AKUISISI DATA DAN PENGOLAHAN SINYAL SPIKE SECARA REAL TIME MENGGUNAKAN PROSESOR 8088 MODE MAKSIMUM

OLEH :

FAIZ RAMADHAN

2882200928

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1994

AKUISISI DATA DAN PENGOLAHAN SINYAL SPIKE SECARA REAL TIME MENGGUNAKAN PROSESOR 8088 MODE MAKSIMUM

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing

Ir. Karyadi, Msc.

SURABAYA

Agustus, 1994

ABSTRAK

Pengambilan event spike membutuhkan proses yang real time dalam artian kapan saja event tersebut timbul dapat diambil dan diproses. Banyak cara yang dapat digunakan dalam mengantisipasi hal tersebut dan sistem paralel adalah salah satu cara yang dapat digunakan.

Proses paralel yang digunakan dalam tugas akhir ini adalah proses *pipeline* karena disamping mudah dalam pemrogramannya juga cocok untuk aplikasi sistem real time.

Prosesor 8088 mode maksimum menawarkan suatu kemungkinan akan aplikasi sistem paralel dengan menggunakan Bus Arbiter 8289 yang bertugas mengatur pemakaian bus bersama sehingga dalam satu waktu hanya ada satu modul yang berhak memakai bus bersama. Selanjutnya penggunaan DMA Controller, ADC Flash dan Co-prosesor 8087 diharapkan dapat menunjang unjuk kerja dari sistem yang dibuat.

Proses yang dilakukan dalam sistem adalah pengambilan data, proses data berupa Filter FIR bandstop dan pengiriman data yang berisi event spike ke komputer lewat serial untuk proses lebih lanjut.

KATA PENGANTAR

Puji syukur Alhamdulillah kami panjatkan ke hadirat Allah SWT, hanya karena Rahmat dan HidayahNya, penulis dapat menyelesaikan Tugas Akhir yang berjudul :

AKUISISI DATA DAN PENGOLAHAN SINYAL SECARA REAL TIME MENGGUNAKAN PROSESOR 8088 MODE MAKSIMUM

Tugas Akhir ini disusun guna memenuhi sebagian persyaratan untuk memperoleh gelar Sarjana Teknik Elektro pada Bidang Studi Elektronika, Jurusan Teknik Elektro, Fakultas Teknik Industri, Institut Teknologi Sepuluh Nopember Surabaya dengan beban 6 SKS (Satuan Kredit Semester).

Dengan selesainya Tugas Akhir ini, dengan hati yang tulus saya mengucapkan terimakasih kepada :

1. Bapak Ir. Karyadi, Msc. selaku dosen pembimbing I yang telah meluangkan banyak waktu untuk memberikan bimbingan kepada penulis hingga selesainya Tugas Akhir ini.
2. Bapak Ir. Harris Pirngadi selaku dosen pembimbing II, atas bimbingan dan tambahan wawasan dalam pengerjaan sampai terselesainya penyusunan Tugas Akhir ini.
3. Bapak Ir. Soetikno selaku Dosen Wali dan Koordinator Bidang Studi Elektronika yang banyak memberikan fasilitas dan bantuan kepada penulis sehingga mampu menyelesaikan tugas akhirnya.
4. Bapak Dr. Ir. Moch. Salehudin, M.Eng.Sc, selaku Ketua Jurusan Teknik

Elektro, beserta seluruh staf pegawai Jurusan Teknik Elektro.

5. Rekan-rekan Bidang Studi Elektronika, terutama rekan-rekan Laboratorium B.203, B.205 dan B.402 yang telah banyak memberikan bantuan dan dorongan selama menyelesaikan Tugas Akhir ini.

6. Bapak dan Ibu yang begitu besar jasanya sehingga tidak mampu ditulis satu persatu.

Akhir kata penulis berharap semoga Tugas Akhir ini bisa bermanfaat bagi seluruh pembaca, khususnya rekan-rekan bidang studi Elektronika.

Surabaya, Juli 1994

Penulis

DAFTAR ISI

DAFTAR GAMBAR

GAMBAR HALAMAN

DAFTAR TABEL

TABEL HALAMAN

BAB I

PENDAHULUAN

I.1 Latar Belakang

Pengambilan event-event cepat seperti spike membutuhkan proses real time dalam artian kapan saja event tersebut muncul dapat diambil dan diproses sehingga tidak ada informasi yang hilang disebabkan pengambilan data tidak setiap saat.

Kemampuan akan pendeteksian event-event cepat dan timbulnya tidak tentu tersebut selanjutnya dapat diaplikasikan untuk mengambil sinyal spike akibat proses transient atau tegangan turun pada jala-jala PLN atau dapat digunakan untuk mengambil event yang disebabkan oleh propagasi pada transmisi data lewat udara.

Jika sistem ini diaplikasikan dilingkungan industri maka dapat ditentukan seberapa besar gangguan yang ditimbulkan oleh kerja peralatan industri yang memakai daya listrik besar. Selanjutnya data yang telah diakuisasi dapat diproses sesuai kebutuhan.

I.2 Permasalahan

Sinyal spike adalah suatu event yang terjadinya sangat cepat dan tidak dapat diketahui secara pasti. Oleh karena itu diperlukan suatu sistem yang dapat bekerja cepat sehingga diharapkan tidak ada event yang tidak terambil disebabkan adanya kelambatan pengambilan dan proses pengolahan data.

I.3 Tujuan

Membuat suatu alat yang dapat mendeteksi event-event seperti spike yang timbulnya tidak dapat diketahui secara pasti dan kemudian disimpan dalam media magnetik untuk dianalisa lebih lanjut.

I.4 Pembatasan Masalah

Pembatasan masalah dalam tugas akhir ini adalah sebagai berikut :

1. Pengambilan sinyal spike dibatasi hanya berupa simulasi dan bukan dari sinyal sesungguhnya mengingat timbulnya event tidak dapat ditentukan.
2. Dalam tugas akhir ini dibatasi hanya mengambil event spike dan tidak menganalisa sebab timbulnya event spike tersebut.

I.5 Metodologi

Langkah-langkah yang diambil untuk mencapai tujuan diatas dapat dibagi menjadi beberapa tahapan, yaitu:

1. Mempelajari metoda-metoda sistem real time yang ada. Selanjutnya mempelajari kemungkinan mengaplikasikan sistem tersebut ke dalam sistem yang berbasis mikro prosesor 8088. Termasuk didalamnya adalah mempelajari komponen-komponen yang berhubungan dengan sistem konversi analog ke digital.
2. Tahap berikutnya adalah mempelajari teori pengolahan sinyal digital yang digunakan untuk memproses sinyal spike yang didapat.
3. Langkah ketiga adalah merancang perangkat keras dari sistem yang

sudah dipelajari diatas dan dilanjutkan dengan perencanaan software yang digunakan untuk pengolahan data.

4. Dari perencanaan yang telah dilakukan, maka dibuat perangkat keras dan perangkat lunak untuk pendeteksi sinyal spike. Hasil yang diperoleh kemudian diuji unjuk kerjanya.
5. Pengujian terhadap alat akan menghasilkan kesimpulan mengenai kemampuan sistem yang dirancang. Akhirnya seluruh langkah-langkah diatas disusun dalam suatu laporan tugas akhir.

I.6 Sistematika

Tugas akhir ini terdiri dari 6 bab dengan penjelasan tiap bab sebagai berikut:

Bab I adalah pendahuluan yang berisi tentang latar belakang, permasalahan, tujuan, metodologi, sistematika dan relevansi.

Bab II mengulas mengenai metode-metode real time termasuk pembahasan proses paralel pipeline, pengenalan prosesor 8088 mode maksimum, DMA Controller, teori tentang Filter Digital FIR.

Bab III membahas bagian-bagian perangkat keras dari sistem yang dibuat yang meliputi sistem modul prosesor 8088 yang mempunyai fasilitas co-prosesor, DMA Controller, dan akses paralel ke interface memory bersama. Disamping itu juga membahas modul ADC yang mentransfer data secara DMA.

Bab IV menjelaskan algoritma dari perangkat lunak yang dipakai.

Bab V berisi kumpulan data-data hasil percobaan yang dilakukan untuk

menguji unjuk kerja dari sistem yang dibuat.

Bab VI adalah penutup yang berisi kesimpulan dan saran-saran kemungkinan pengembangan dari sistem yang telah dibuat.

I.7 Relevansi

Dengan mendeteksi timbulnya noise pada suatu daerah maka dalam pengolahan data selanjutnya dapat diperkirakan secara spesifik intensitas noise di suatu daerah industri.

Sistem pendeteksi sinyal spike ini dibuat dalam bentuk modul-modul yang dapat bekerja secara paralel. Oleh karena itu dapat digunakan untuk penelitian selanjutnya dalam bidang pemrograman secara paralel. Setiap modulnya berdiri sendiri dan secara terpisah dapat digunakan untuk aplikasi lain.

BAB II

TEORI PENUNJANG

II.1 Konfigurasi Paralel pada 8088 Mode Maksimum

II.1.1 Tinjauan umum

Sistem konfigurasi paralel merupakan salah satu bidang penelitian yang semakin berkembang. Hal ini ditunjang dengan semakin murahnya unit prosesor dan semakin berkembangnya teknologi Integrated Circuit.

Dewasa ini semakin banyak dikembangkan sistem elektronik yang mengacu pada kecepatan tinggi. Salah satu sistem yang dikembangkan saat ini adalah sistem konfigurasi paralel beberapa prosesor. Sistem paralel ini sendiri sudah sedemikian berkembang sehingga diciptakan suatu konsep pemrograman paralel (*Concurrent Programming*).

Dengan dipakainya konfigurasi beberapa prosesor secara paralel diharapkan unjuk kerja yang dihasilkannya akan meningkat pula.

Prosesor 8088 adalah prosesor keluaran Intel yang sudah begitu banyak digunakan pada komputer dan software pendukungnya pun sudah sedemikian banyak sehingga pemakaiannya dalam proses paralel akan menambah wawasan cara memperoleh sistem yang real time.

Ada tiga cara mengimplementasikan proses paralel, yaitu:

1. *Multiprogramming* adalah pelaksanaan proses paralel pada komputer dengan satu prosesor. Contoh dari istilah ini adalah pemrograman *multitasking*.

2. *Multiprocessing* adalah pelaksanaan proses paralel pada beberapa prosesor dan tiap prosesor berkomunikasi lewat suatu memori bersama. (Tiap prosesor bisa mempunyai memory lokal yang hanya bisa diakses oleh prosesor itu saja).
3. *Distributed processing* adalah pelaksanaan proses paralel seperti pada *Multiprocessing* hanya saja dalam hal ini komunikasi antar prosesor tidak melalui memory bersama melainkan lewat pertukaran pesan ke memory lokal tiap prosesor. Jadi dalam hal ini tidak ada memory bersama.

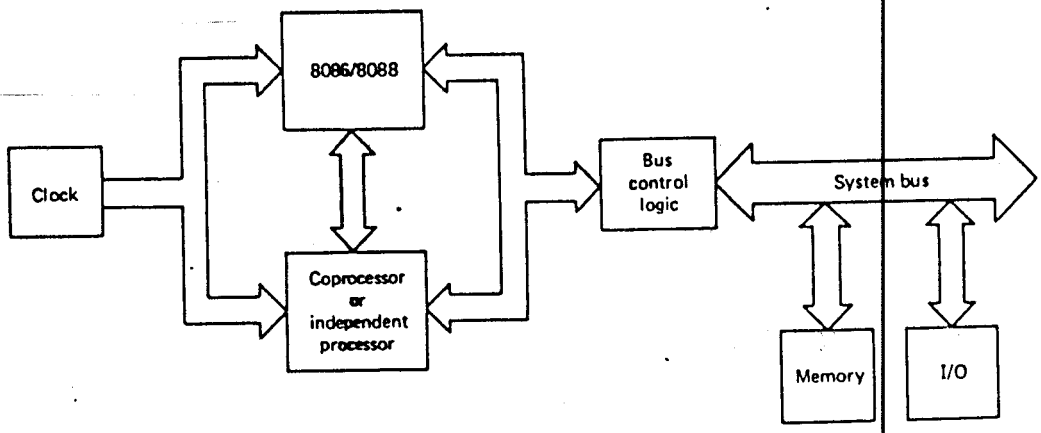
II.1.2 Konfigurasi *Multiprocessor* pada 8088 mode maksimum

Mode maksimum dari 8088 secara khusus dapat dipakai untuk konfigurasi multi prosesor. Ada tiga konfigurasi dasar dari sistem multi prosesor, yaitu :

1. Konfigurasi *Coprocessor*
2. Konfigurasi *Closely Coupled Processor*
3. Konfigurasi *Loosely Coupled Processor*

1. Konfigurasi *Co-processor*

Blok diagram dari konfigurasi ini terlihat seperti pada gambar 2.1. Dalam konfigurasi ini ada prosesor yang bertindak sebagai *master* dan prosesor yang lain sebagai *slave*. Kedua prosesor ini secara bersama-sama menggunakan memory, data bus, control bus, pembangkit *clock* dan sistem I/O. Prosesor slave dalam operasinya tidak dapat bekerja sendiri dan selalu tergantung pada prosesor master.



Gambar 2.1 Konfigurasi coprocessor dan Closely Coupled[1]

2. Konfigurasi *Closely Coupled*

Konfigurasi ini secara sistem tidak banyak berbeda dengan konfigurasi Co-processor. Hanya saja dalam hal ini prosesor yang bekerja sebagai slave dapat bekerja sendiri dan tidak bergantung pada prosesor master.

Dua prosesor 8088/8086 tidak dapat bekerja pada dua model konfigurasi ini, karena sifat prosesor 8088/8086 yang selalu bertindak sebagai prosesor master.

3. Konfigurasi *Loosely Coupled*

Konfigurasi ini digunakan pada sistem medium sampai kompleks. Setiap modul dalam sistem bisa bertindak sebagai master bus dan tiap modul terdiri dari satu prosesor master dan satu atau lebih prosesor dalam konfigurasi co-processor

---

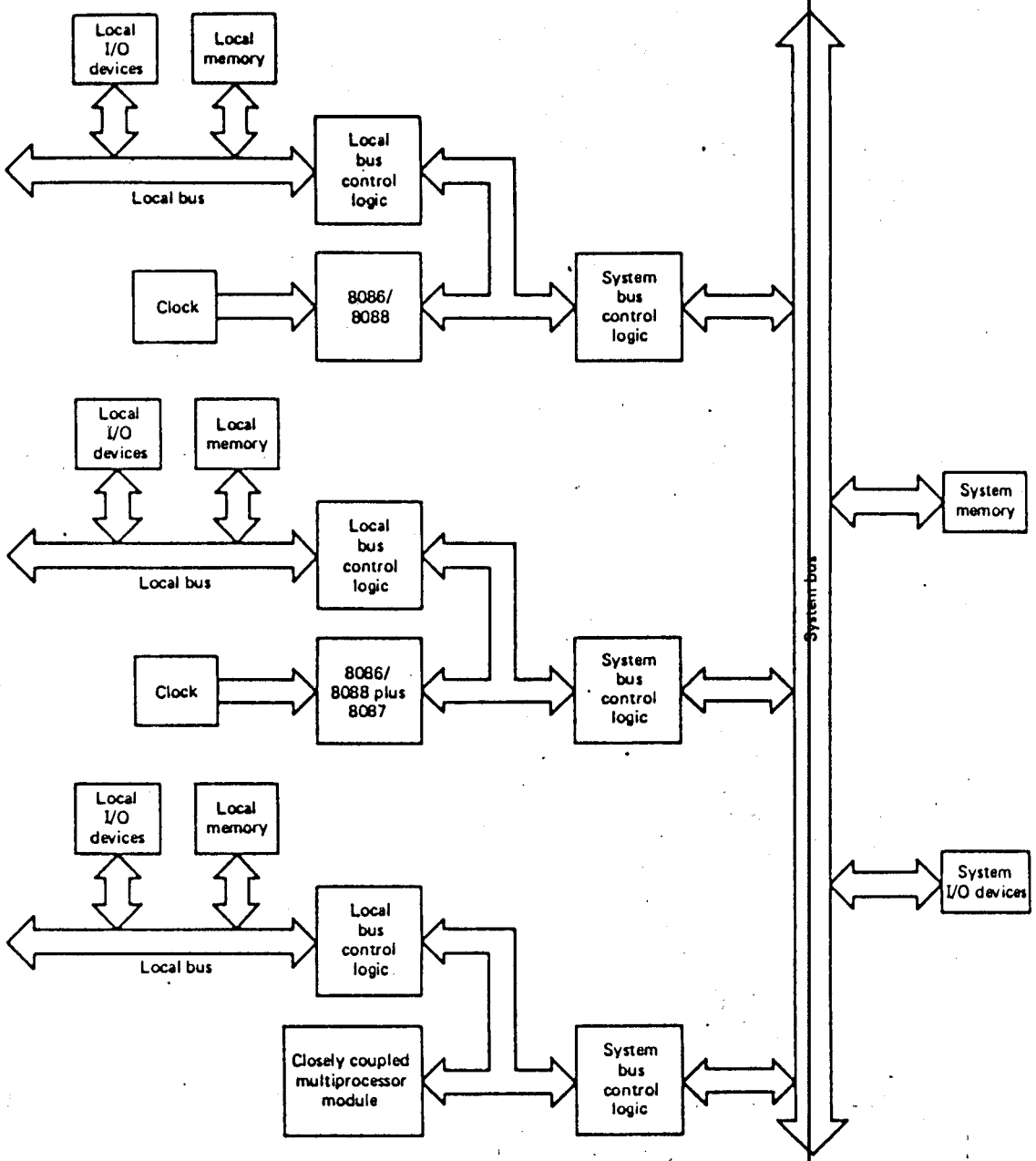
[1]Liu, Yu-Cheng, *Microcomputer Systems : The 8086/8088 Family : Architecture, Programming, and Design, Second Edition*, Prentice Hall of India, hal. 451

atau closely coupled. Modul-modul tersebut dapat mengakses sumber bersama (*common resources*) dan suatu rangkaian logic diperlukan untuk menghindari pemakaian bus secara bersama dalam satu waktu. Seperti terlihat pada gambar 2.2, setiap modul bekerja sendiri-sendiri dan tidak ada hubungan langsung antar modul. Komunikasi antar modul dapat dilakukan melalui sumber bersama. Disamping sumber-sumber bersama, tiap modul bisa mempunyai memory lokal dan peralatan I/O sendiri. Tiap prosesor dalam modul terpisah secara simultan dapat menjalankan suatu proses sendiri melalui lokal data bus, sehingga jelas meningkatkan kecepatan proses paralel.

Dalam tugas akhir ini digunakan konfigurasi *Loosely Coupled*, karena konfigurasi ini mempunyai beberapa keuntungan, yakni :

1. Kecepatan proses yang tinggi dapat dicapai dengan mempekerjakan beberapa CPU sekaligus.
2. Sistem ini dapat diperlebar dalam bentuk modular. Setiap modul berdiri sendiri dan biasanya terpisah dalam PCB sendiri. Sebab itu, satu modul dapat dipasang atau dilepas tanpa mempengaruhi modul lainnya.
3. Kerusakan pada satu modul biasanya tidak menyebabkan kerusakan sistem secara keseluruhan dan modul yang rusak dapat dengan mudah dideteksi dan diganti.
4. Setiap master bus bisa mempunyai bus lokal untuk mengakses lokal memory dan peralatan I/O.

Dalam sebuah sistem loosely coupled multi prosesor, lebih dari satu modul



Gambar 2.2 Konfigurasi *Loosely Coupled*[2]

bisa mengakses sumber bersama. Oleh karena itu diperlukan suatu rangkaian logic tambahan untuk mengatur pemakaian bus bersama, sehingga hanya satu modul

[2]Ibid, hal. 452

yang bisa mengakses bus pada satu waktu. Permintaan bus yang bersamaan dipecahkan dengan menggunakan prioritas.

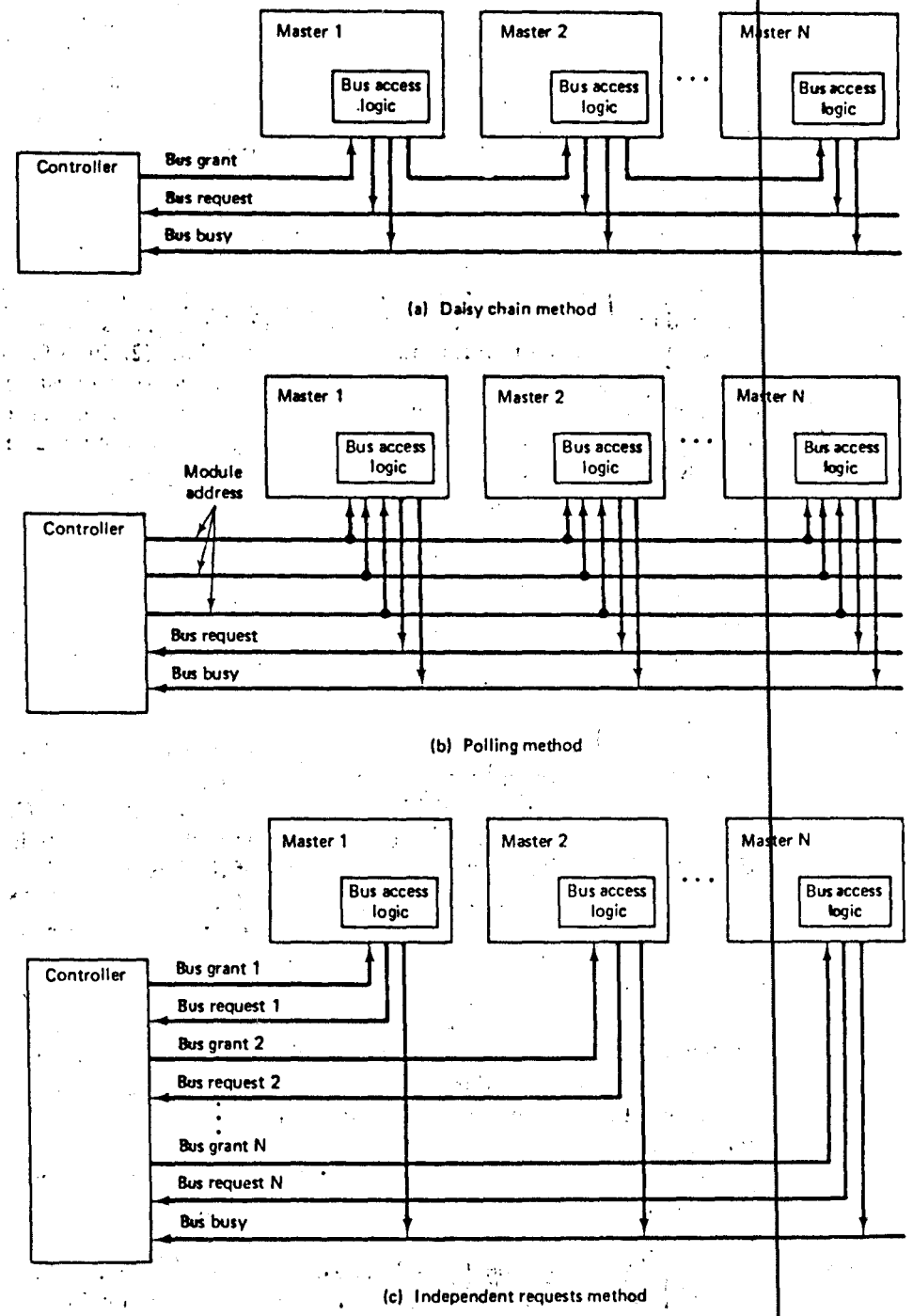
Ada tiga metoda prioritas yang bisa digunakan, yaitu:

1. Metode *Daisy chaining*
2. Metode *Polling*
3. Metode *Independent requesting*

1. Metode *Daisy Chaining*

Metode ini terkenal karena mudah dan murah. Semua modul menggunakan jalur yang sama untuk mengaktifkan sinyal bus request seperti terlihat pada gambar 2.3(a). Terlihat bahwa modul yang letaknya paling dekat dengan controller mempunyai prioritas tertinggi.

Dibanding dengan dua metode yang lain, metode daisy chain adalah yang paling sedikit menggunakan sinyal kontrol dan besarnya tidak tergantung pada jumlah modul yang digunakan. Waktu arbitrasi untuk metode ini lambat disebabkan oleh delay propagasi dari sinyal bus grant. Lamanya waktu delay ini berbanding lurus dengan jumlah modul. Oleh karena itu modul pada sistem dengan metode daisy chain terbatas jumlahnya. Disamping itu prioritas tiap modul sudah tertentu dan jika ada kerusakan pada satu modul menyebabkan seluruh sistem terhenti.



Gambar 2.3 Skema metoda prioritas pemakaian bus[3]

[3]Ibid, hal. 465

2. Metode *Polling*

Metode ini menggunakan jalur address untuk menandai tiap modul. Blok diagramnya adalah seperti terlihat pada gambar 2.3(b). Setelah menerima sinyal bus request, controller mengaktifkan address modul secara berurutan. Modul yang mengeluarkan sinyal request akan mengaktifkan sinyal busy setelah menerima address dari controller. Keuntungannya adalah prioritas tiap modul dapat diubah dengan mudah.

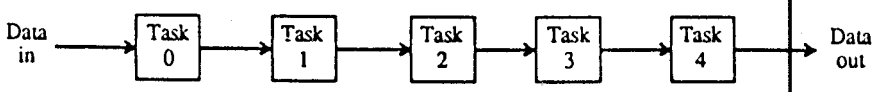
3. Metode *Independent Request*

Metode ini memakai prioritas secara paralel, yaitu tiap modul mempunyai sepasang sinyal bus request dan bus grant dan jalur prioritas seperti terlihat pada gambar 2.3(c). Controller menggunakan dekoder prioritas yang akan menentukan request mana yang tertinggi. Pada mode ini waktu arbitrasi adalah cepat dan tidak tergantung pada jumlah modul. Metode ini paling cepat jika dibanding dengan dua metode yang lain. Tetapi jumlah jalur bus request dan bus grant adalah sebanyak 2 kali jumlah modul.

II.1.3 Proses *Pipeline*

Proses pipeline adalah proses kerja paralel beberapa prosesor terhadap suatu data elemen. Data elemen tersebut diproses secara bergiliran oleh beberapa prosesor dimana tiap prosesor mempunyai tugas berbeda satu dengan yang lain. Metode ini bisa dimasukkan dalam katagori algoritma paralel karena secara algoritma proses terdistribusi pada sekelompok prosesor. Sebaliknya jika proses

yang sama terdapat pada tiap prosesor dan data yang diproses berbeda pada tiap prosesor maka metode ini termasuk katagori data paralel. Blok diagram proses pipeline adalah seperti pada gambar 2.4.



Gambar 2.4 Proses pipeline sekelompok prosesor[4]

Proses pipeline termasuk metode paralel yang mudah dalam aplikasinya dibanding dengan sistem yang lain. Proses ini mudah dimengerti, dibangun, dikontrol dan diprogram. Teknik komunikasinya pun juga mudah dikembangkan dan programnya seperti program sekuensial biasa. Karena itu proses pipeline biasanya digunakan pada sistem realtime.

Efisiensi proses pipeline

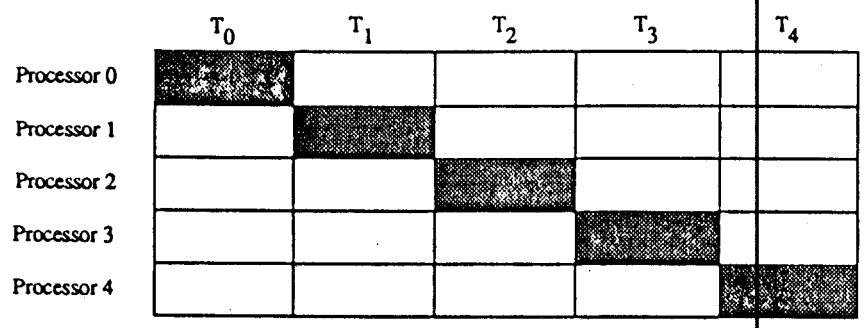
Efisiensi proses pipeline terutama sekali tergantung pada pendistribusian program atau task pada beberapa prosesor. Program harus dibagi sebanyak jumlah modul prosesor yang ada dan idealnya setiap prosesor seharusnya memperoleh jumlah tugas yang sama karena jika ada salah satu prosesor mempunyai tugas yang lebih lama maka akan ada waktu tunggu timbul pada prosesor berikutnya sehingga akan berpengaruh atas efisiensi sistem.

Setiap data yang diproses dapat dianggap sebagai suatu paket data. Jika paket data yang diproses pada suatu waktu adalah satu maka diagram aktifitasnya

---

[4]Cok, Ronald S., ***Paralel Programs for the Transputer***, Prentice Hall, hal. 90

adalah seperti pada gambar 2.5. Setiap prosesor akan bekerja secara bergantian, tetapi karena hanya ada satu paket data maka hanya ada satu prosesor yang bekerja pada satu interval waktu.



Gambar 2.5 Diagram aktifitas lima prosesor untuk satu paket data[5]

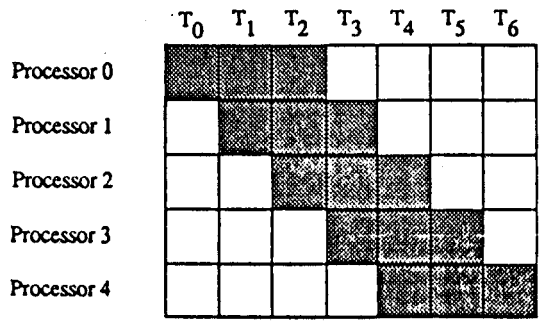
Pada gambar 2.5 ditunjukkan aktifitas prosesor terhadap waktu. Kotak yang berarsir menunjukkan kerja prosesor pada periode waktu tersebut dan kotak yang tidak berarsir menunjukkan prosesor tidak bekerja pada waktu itu. Efisiensi dari proses adalah 20%, karena hanya satu prosesor yang bekerja dari lima yang tersedia pada satu interval waktu (lima interval waktu terpakai dari 25 interval waktu tersedia). Jika n prosesor yang dipakai, maka efisiensinya adalah 1/n.

Jika data yang diproses ada tiga paket maka efisiensinya bertambah, seperti terlihat pada gambar 2.6. Dalam hal ini, tiap prosesor bekerja selama tiga periode waktu sehingga waktu yang terpakai bertambah yaitu 15 dari 35 tersedia. Efisiensinya pun juga meningkat menjadi 43%.

Jumlah interval waktu yang dibutuhkan untuk menyelesaikan suatu proses

[5]Ibid, hal. 92

akan tergantung pada jumlah paket data yang diproses dan jumlah prosesor yang



Gambar 2.6 Diagram aktifitas lima prosesor untuk tiga paket data[6]

bekerja. Jika kita mengabaikan waktu overhead komunikasi antar prosesor, dengan N menunjukkan jumlah paket data, P jumlah prosesor dan T adalah waktu proses terlama dari task-task yang ada, maka waktu total proses adalah :

$$T_{total} = (N + (P - 1)) \; x \; T_{proses\ task\ terlama} \qquad (2.1)$$

Efisiensi keseluruhan dari sistem pipeline didefinisikan sebagai waktu yang dibutuhkan untuk mengerjakan task dengan satu prosesor dibagi waktu yang digunakan proses pipeline mengerjakan task dikali jumlah prosesor. Waktu proses yang diperlukan oleh proses pipeline adalah seperti pada persamaan 2.1. Dengan N menyatakan jumlah paket data, P jumlah prosesor dan T adalah waktu proses task terlama maka efisiensi keseluruhan adalah :

$$E = \frac{T_{total\ dengan\ satu\ prosesor}}{P \; x \; (N + P - 1) \; x \; T_{proses\ task\ terlama}} \qquad (2.2)$$

[6]Loc. cit.

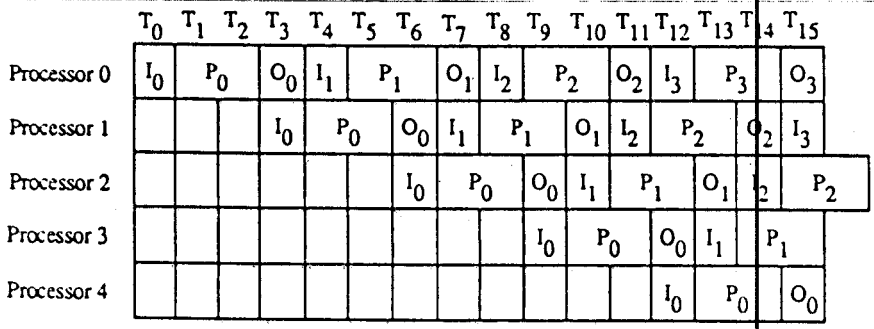
Metode komunikasi antar prosesor

Metode komunikasi yang dikembangkan disini sangat berpengaruh terhadap waktu overhead komunikasi antar prosesor yang berarti juga akan berpengaruh terhadap efisiensi keseluruhan. Ada beberapa teknik metoda komunikasi yang dikembangkan tetapi dalam tugas akhir ini hanya akan dibahas satu metoda yang dipakai yaitu metode **single buffering.**

Metode single buffering

Metode ini adalah yang termudah dari semua metode komunikasi yang dikembangkan. Diagram aktifitasnya adalah seperti pada gambar 2.7. Dengan metode ini tiap prosesor mempunyai suatu lokasi memori untuk menyimpan data. Pertama prosesor mengambil data, memprosesnya dan selanjutnya mengirimnya ke prosesor lain. Sehingga total waktu proses untuk suatu prosesor adalah penjumlahan dari waktu yang dibutuhkan untuk mengerjakan proses ambil data, proses dan output data ke prosesor lain (memori bersama).

Seperti terlihat pada gambar 2.7, tiap baris menunjukkan aktifitas dari prosesor yang bersangkutan dalam suatu waktu. Pada setiap kotak, I menunjukkan input, P menunjukkan proses, O menunjukkan output dan kotak kosong menunjukkan tidak ada aktifitas; angka indeks menunjukkan nomor paket data yang sedang diproses. Tiap prosesor secara bergantian mengambil input data, memprosesnya dan mengeluarkan outputnya, sehingga seperti terlihat pada gambar 2.7 kecepatan proses pipeline adalah 4 interval waktu. Juga terlihat dari diagram bahwa proses tidak akan mengeluarkan output paket data pertama sebelum interval waktu ke 15.



Gambar 2.7 Diagram aktifitas proses pipeline single buffer dengan 5 prosesor[7]

Kelimabelas interval waktu tersebut adalah waktu delay laten (*latency*) dari proses pipeline yang merupakan waktu diperlukan untuk memproses satu paket data.

II.1.4 Prosesor 8088 Mode Maksimum

Jika pin 33 MN/MX dihubungkan ke ground maka prosesor 8088 akan bekerja pada mode maksimum. Tabel 2.1 dibawah menerangkan pin 8088 khusus pada mode maksimum. Pin lainnya sama dengan mode minimum dan tidak diterangkan disini. Dari tabel 2.1 jelas sekali terlihat bahwa perbedaan utama mode maksimum dan minimum adalah diperlukannya rangkaian logic tambahan untuk menerjemahkan status S0, S1 dan S2 menjadi sinyal kontrol. Dalam hal ini tugas tersebut ditangani oleh bus controller 8288.

II.1.5 Bus Controller 8288

Hubungan 8288 ke 8088 secara lebih jelas adalah seperti pada gambar 2.8 Pin S0\, S1\ dan S2\ 8288 merupakan input status dari 8088 dan selanjutnya

---

[7]Ibid, hal. 98

didekodekan oleh 8288 menjadi perintah yang diinginkan (memori read-write atau

Tabel 2.1 Pin 8088 khusus pada mode maksimum

Pin	Simbol	In/Out 3-State	Penjelasan
24,25	QS1,QS0	Output	Menyatakan status dari urutan instruksi yang menyatakan aktifitas dari urutan selama satu cycle (4 clock) sebelumnya.
26,27,28	S0,S1,S2	Output 3-State	Menyatakan transfer yang akan dilakukan pada cycle tersebut. S0 S1 S2 0 0 0 Interrupt Acknowledge 0 0 1 Baca port I/O 0 1 0 Tulis port I/O 0 1 1 Halt 1 0 0 Baca instruksi 1 0 1 Baca memory 1 1 0 Tulis memory 1 1 1 Passive, tidak ada kegiatan
29	LOCK	Output 3-State	Menyatakan bahwa bus tidak diserahkan kepada modul master lain. Ini ditandai dengan prefiks instruksi LOCK dan tetap aktif sampai akhir instruksi berikutnya.
30	RQ/GT1	I/O	Input dari sinyal Bus Request dan output dari sinyal Bus Grant
31	RQ/GT0	I/O	Sama seperti RQ/GT1 tetapi dengan prioritas lebih tinggi.

atau I/O read-write). Pin ALE, DT/R dan DEN mempunyai fungsi yang sama dengan output yang dikeluarkan 8088 pada mode minimum (kecuali DEN yang merupakan kebalikan dari output DEN\ mode minimum). Pin AEN\, IOB dan

CEN adalah pin yang digunakan khusus untuk sistem prosesor banyak (*multiprocessor*). Dalam gambar 2.8 ditunjukkan prosesor bekerja hanya dalam sistem prosesor tunggal (*single processor*) sehingga AEN\ dan IOB dihubungkan ke ground dan CEN diset ke logic satu.

Pin IOB (diset high) berfungsi untuk mengeset 8288 dalam mode bus I/O yaitu output perintah ke I/O tidak bisa dikontrol oleh AEN\, yaitu pin yang jika nonaktif (high) membuat output 8288 dalam keadaan high impedance. Sedangkan pin CEN jika diset high output perintah dapat keluar dan jika nonaktif (diset low), output perintah 8288 akan nonaktif (selalu high tetapi tidak dalam keadaan high impedance).

Output pin MCE\PDEN tergantung pada mode yang diset pada pin IOB. Jika IOB diset low, output berarti Master Cascade Enable (MCE) yang digunakan untuk mengatur 8259 kaskade. Jika IOB diset high maka outputnya adalah PDEN yang digunakan pada sistem dengan konfigurasi banyak bus (*multibus*).

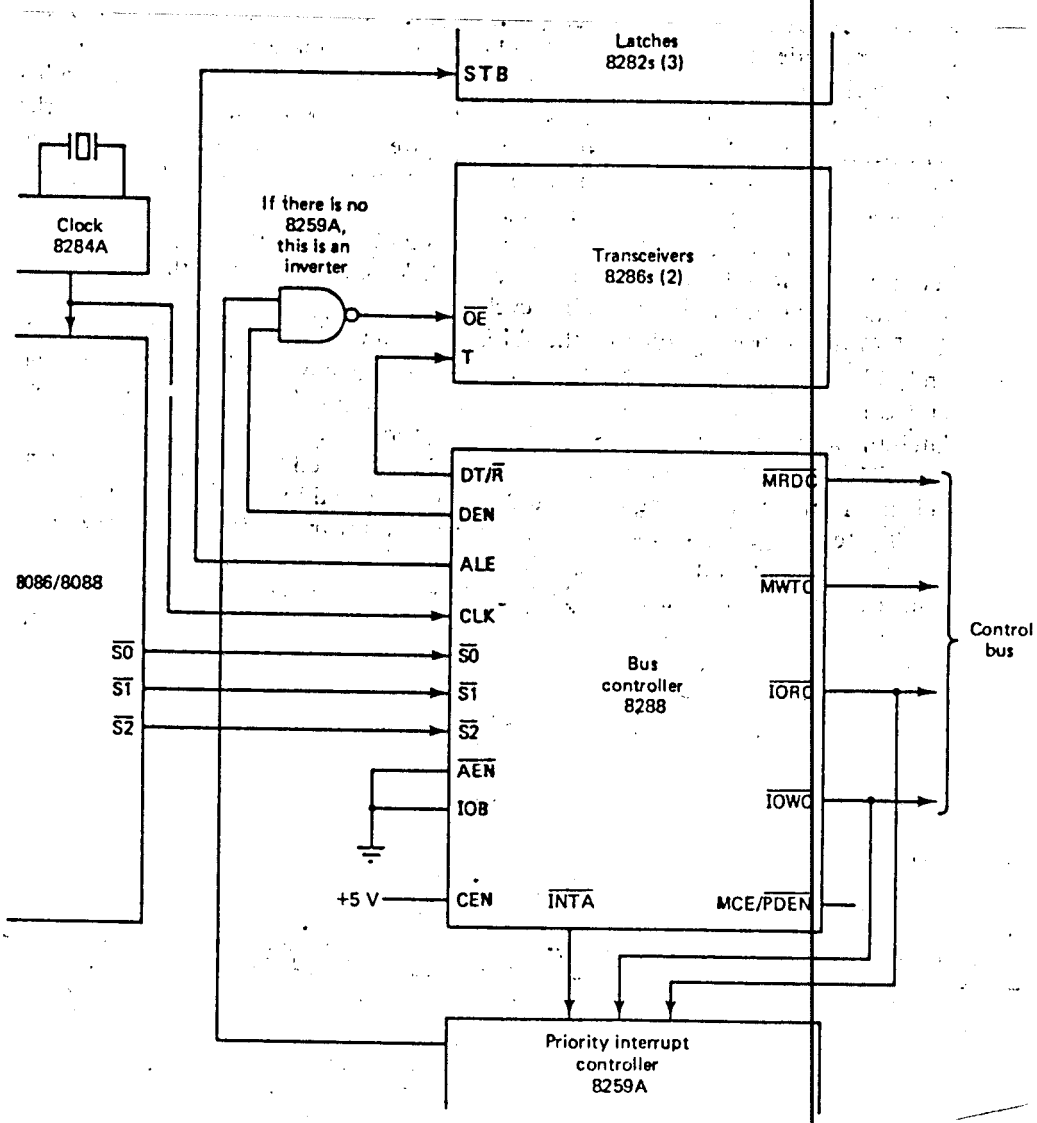
Selanjutnya pin lain pada 8288 dalam gambar 2.8 mempunyai fungsi atau definisi sebagai berikut:

INTA mengeluarkan sinyal *interrupt acknowledge* dua kali ke PIC atau ke peralatan lain yang mengeluarkan interrupt.

IORC (*I/O Read Command*) berfungsi mengambil data dari I/O.

IOWC (*I/O Write Command*) berguna untuk mengirim data ke port dengan alamat yang telah didefinisikan.

MRDC (*Memory Read Command*) memerintahkan memori meletakkan data dari lokasi tertentu di data bus.



Gambar 2.8 Sambungan 8288 ke 8088

MWTC (*Memory Write Command*) mengirim data di data bus ke lokasi memori dengan alamat tertentu.

Pin lain yang tidak masuk pada gambar 2.8 adalah AIOWC\ (*Advanced I/O write command*) dan AMWC\ (*Advanced Memory Write*). Fungsi pin-pin ini sama dengan IOWC\ dan MWTC\ hanya saja mereka diaktifkan satu clock lebih

cepat. Biasanya digunakan pada peralatan yang lambat sehingga ada waktu ekstra untuk menyiapkan data.

II.1.5 Bus arbiter 8289

8289 bus arbiter bekerja bersama-sama dengan bus controller 8288 agar prosesor 8088 dapat bekerja secara paralel. Dalam operasinya, mikroprosesor tidak mengetahui keberadaan dari bus arbiter ini dan mengeluarkan perintah seolah-olah ia mempunyai hak eksklusif atas bus sistem. Jika prosesor tidak memegang kontrol bus sistem multimaster, maka arbiter menonaktifkan bus controller, bus driver (data transceiver dan address latches), yaitu dengan membuat outputnya menjadi high impedance. Karena perintah tidak dikeluarkan oleh 8288, maka bus sistem akan tampak seperti tidak siap (*"Not Ready"*) dan prosesor akan memasuki keadaan tunggu (*wait state*). Prosesor akan tetap dalam keadaan ini sampai 8289 memperoleh hak pemakaian dari bus sistem multimaster. Selanjutnya 8289 mengaktifkan bus controller dan output driver sehingga prosesor dapat melanjutkan dan menyelesaikan tugasnya.

Dibawah ini adalah penjelasan mengenai fungsi pin-pin penting dari Arbiter 8289:

1. RESB — **Resident Bus** : adalah pin yang membuat arbiter bekerja dalam mode bus residen. Jika diberi input high, maka pemakaian atas bus multimaster tergantung dari input pin SYS/(RESB\). Jika diberi input low pin SYS/(RESB\) tidak berfungsi.

2. ANYRQST **Any Request** : input high ke pin ini memungkinkan pemakaian bus multimaster berikutnya diberikan ke arbiter yang mempunyai prioritas lebih rendah.

3. IOB\ **IO Bus** : input low ke pin ini akan membuat arbiter bekerja dalam mode bus IO.

4. AEN\ **Address Enable** : sinyal output dari arbiter yang berfungsi untuk menonaktifkan output address latch prosesor, bus controlller 8288.

5. SYSB/(RESB\) **System Bus/Resident Bus** : merupakan input bagi arbiter apakah prosesor hendak menggunakan bus multimaster atau bus lokal (Resident Bus). Jika input high maka arbiter akan meminta pemakaian bus multimaster. Jika input low maka kontrol atas bus multimaster dilepaskan.

6. BREQ\ **Bus Request** : output aktif low yang menandakan arbiter meminta kontrol atas bus multimaster.

7. BPRN\ **Bus Priority In** : sinyal input aktif low yang menyatakan bahwa arbiter dapat memegang kontrol atas bus multimaster.

8. BPRO\ **Bus Priority Out** : output aktif low yang berfungsi memberikan prioritas pemakaian bus ke arbiter dengan prioritas berikutnya dalam metoda prioritas daisy chain.

Ada dua mode operasi dari 8289, yaitu:

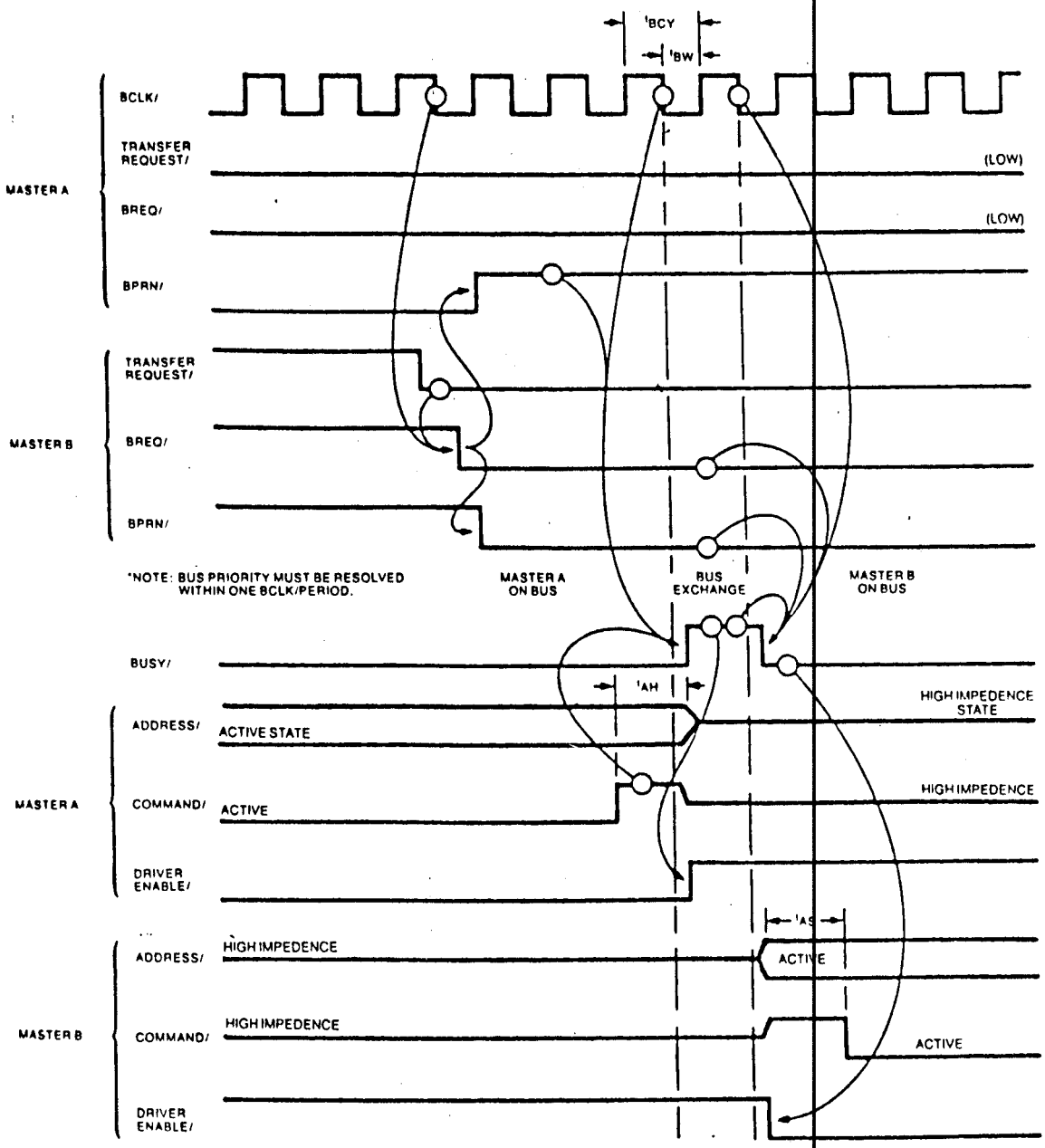
1. **Mode Bus I/O.**

   Dalam mode ini prosesor mengakses dua bus, yaitu bus sistem multimaster dan I/O bus. I/O bus adalah bus dimana semua peralatan pada bus tersebut termasuk memory diperlakukan sebagai peralatan I/O dan dialamati menggunakan perintah I/O. Sedangkan semua perintah memory digunakan untuk mengakses bus sistem multimaster. Agar dapat bekerja pada mode ini maka pin IOB dan pin RESB dibuat low.

2. **Mode Bus Resident.**

   Dalam mode ini prosesor memiliki dua bus, yaitu bus Resident dan bus sistem multimaster. Bus Resident dalam hal ini dapat mengeluarkan perintah memory dan I/O yang terpisah dari perintah untuk bus sistem multimaster. Bus resident hanya bisa melayani satu master dan siap digunakan kapan saja. 8289 akan bekerja dalam mode ini dengan memberi input high ke pin RESB dan IOB.

Adapun diagram waktu perpindahan pemakaian atas bus sistem master dari arbiter prioritas lebih rendah ke arbiter prioritas lebih tinggi dapat dilihat pada gambar 2.9.



Gambar 2.9 Diagram waktu pengambilalihan kontrol pemakaian bus sistem multimaster[8]

II.2 Math Co-Processor 8087

Untuk beberapa aplikasi tertentu, dibutuhkan pengolahan data yang berupa perhitungan yang cepat. Adanya penundaan waktu akibat proses perhitungan yang

[8]Hall, Douglas V., *Microprocessors and Interfacing : Programming and Hardware*, McGraw-Hill Book Co., hal.403

lambat dapat menyebabkan data yang diperoleh tidak memberikan hasil yang memuaskan. Oleh karena itu prosesor 8088 dirancang untuk dapat dipakai bersama-sama dengan math co-processor 8087 yang mempunyai kemampuan perhitungan cepat, yaitu suatu perhitungan jika dilakukan dengan prosesor 8088 memerlukan beberapa instruksi maka dengan adanya 8087 ini perhitungan tersebut dapat dikerjakan hanya dengan satu instruksi 8087 sehingga diperoleh kecepatan proses yang lebih tinggi.

Dalam operasinya 8087 bersama-sama dengan 8088 melakukan pembacaan instruksi dari memori. Tetapi 8087 hanya melakukan instruksi yang memang ditujukan untuknya dan mengabaikan instruksi untuk 8088. Sebaliknya 8088 yang juga membaca instruksi 8087 yang berupa instruksi *Escape*, juga mengabaikannya tetapi tetap melakukan pembacaan terhadap operand-operandnya. Operand ini kemudian dipakai oleh 8087 untuk melakukan pembacaan atau penulisan data dari atau ke memori.

Untuk menjamin terjadinya sinkronisasi dari 8087 dan 8088 yang bekerja secara paralel, maka program-program harus mengikuti aturan sebagai berikut :

- 8088 tidak boleh melakukan perubahan terhadap memori yang dipakai dalam instruksi 8087 sampai 8087 selesai mengerjakan instruksi tersebut.
- Instruksi 8087 berikutnya tidak boleh dibaca oleh 8088 sampai instruksi yang dikerjakan oleh 8087 selesai dilakukan.

Sistem bilangan

8087 mengenal 3 sistem bilangan, yaitu : floating point, integer dan *Packed Decimal*. Dari ketiga sistem tersebut yang terpenting adalah floating point.

Floating point merupakan sistem bilangan bagi komputer yang serupa dengan bentuk eksponen. Sebagai contoh nilai "negatif setengah" dapat ditulis dalam bentuk eksponen : -5.0×10^{-1}. Tiga bagian penting dari bentuk tersebut adalah *sign* (tanda negatif/positif), *mantissa* (angka penting) dan *eksponen*

Tabel 2.2 memberikan jumlah bit yang dipakai untuk menyimpan suatu jenis data, jumlah angka penting serta range untuk setiap jenis data.

Tabel 2.2 Jenis data pada 8087[9]

Jenis data	Jumlah bit	Angka penting	Range
Word Integer	16	4	-32768 s/d 32767
Short Integer	32	9	-2×10^{9} s/d 2×10^{9}
Long Integer	64	18	-2×10^{18} s/d 2×10^{18}
Packed Decimal	80	18	18 angka dec + sign
Short Real	32	6 / 7	10^{-37} s/d 10^{38}
Long Real	64	15 / 16	10^{-307} s/d 10^{308}
Temporary Real	80	19	10^{-4932} s/d 10^{4932}

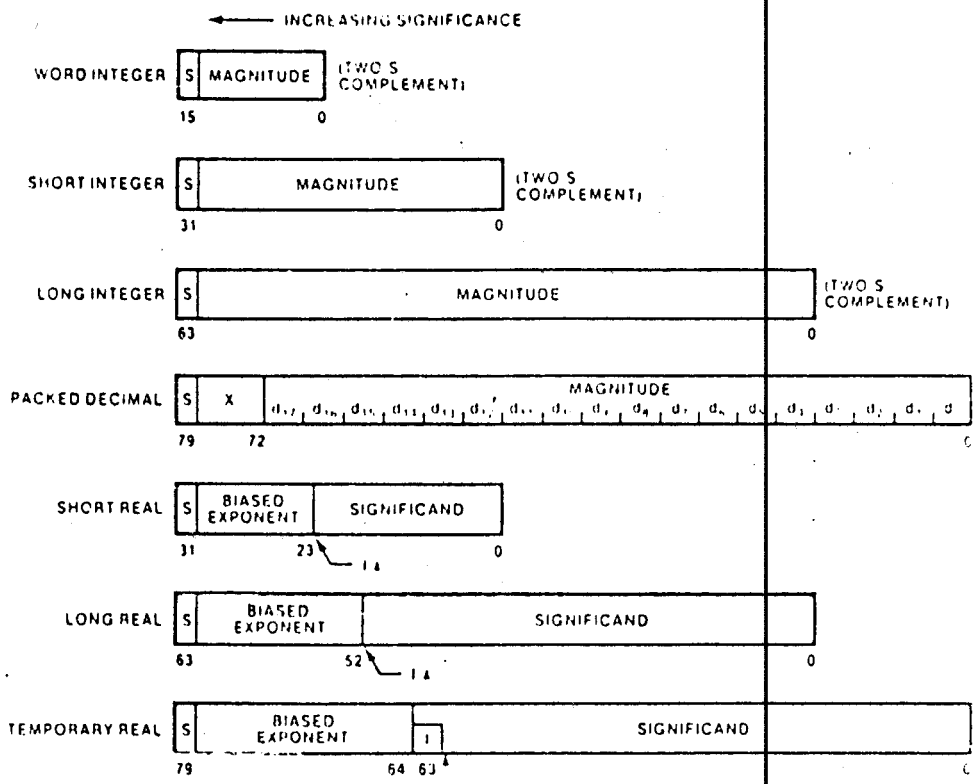
Format data pada 8087

Gambar 2.10 memperlihatkan fungsi tiap bit data pada 8087 dan cara menyimpan bit tersebut pada memori adalah sama seperti cara prosesor 8088 menyimpan word, double word dan seterusnya.

Untuk data floating point, 8087 mempunyai cara khusus dalam menyimpan nilai-nilai bitnya. Berikut adalah aturan penyimpanan nilai bit untuk floating point.

---

[9]Startz, Richard, *8087 Application and Programming for the IBM PC and Other PCs*, Prentice Hall Publishing, California 1983, hal. 30



Keterangan :
S = bit tanda (sign), 0 = positif, 1 = negatif
X = don't care (diabaikan)
Dn = angka desimal (1 nibble)

Gambar 2.10 Format bit untuk tiap jenis data pada 8087[10]

- Pendefinisian bilangan float dinyatakan dalam binary biasa dan bukan packed decimal.
- Angka Floating point selalu dinyatakan dalam bentuk "ternormalisir". Bit awal dari angka tersebut selalu satu sehingga tidak dimasukkan dalam data. Untuk memperoleh angka yang ternormalisir, 8087 akan menggeser angka penting (*significant*) sementara mengurangi atau menambah nilai eksponen.

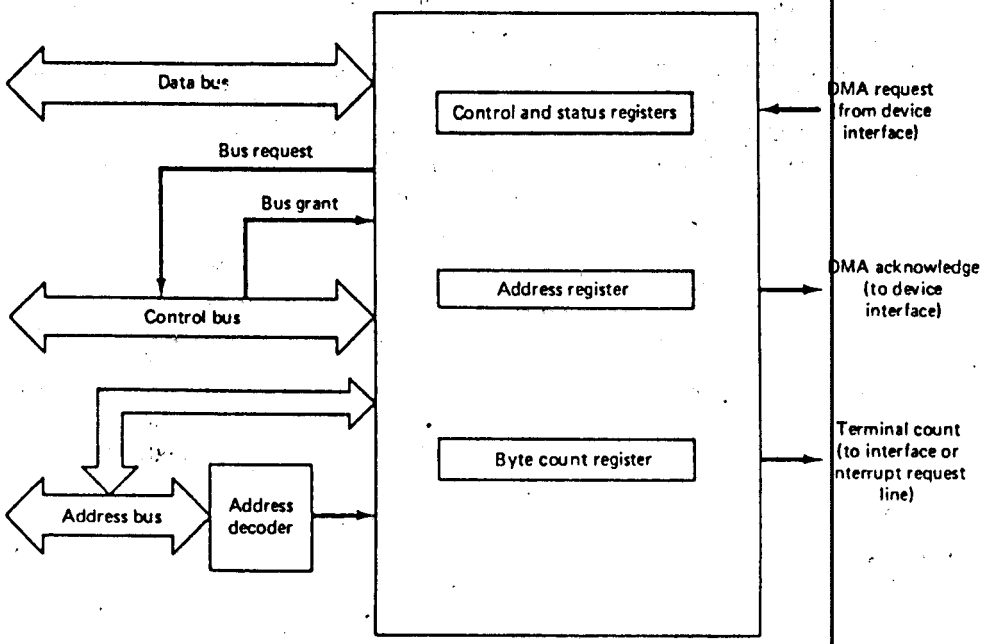
[10]Ibid, hal. 33

- Eksponen dapat bernilai positif atau negatif. Untuk menyatakan ini komputer tidak menggunakan sign bit, melainkan langsung menjumlahkan dengan suatu konstanta basis. Untuk angka real single precision nilai eksponen sesungguhnya ditambah dengan 127 sebelum disimpan.
- Angka nol dinyatakan dengan semua bit eksponen serta bit significant dibuat nol. Bit tanda diabaikan.

II.3 DMA Controller 8237-5

Jika kecepatan transfer data dari atau ke I/O relatif lambat, maka digunakan metoda interrupt untuk transfer data per byte atau word. Tetapi pelaksanaan interrupt kadang memakan waktu lebih lama dari waktu yang tersedia. Sebagai contoh, sebuah unit disk magnetik mengirim data ke komputer dengan kecepatan 200.000 byte per detik, yang berarti dibutuhkan waktu kurang dari 5 mikro detik untuk mengirimkan satu byte data ke memory. Untuk tingkat transfer data secepat ini metode interrupt tidak dapat digunakan dan dalam hal ini sebuah DMA controller diperlukan.

DMA controller adalah suatu alat yang mampu melakukan transfer antara I/O (tempat penyimpanan data) dengan memory tanpa melibatkan kerja CPU. Dalam hal ini transfer dilakukan diantara dua siklus kerja dari mikroprosesor. Organisasi dari satu channel DMA beserta hubungannya secara umum dapat dilihat pada gambar 2.11.



Gambar 2.11 Organisasi DMA Controller secara umum[11]

Konsep dasar transfer data dari DMA Controller

Selama pelaksanaan program secara normal, sistem bus termasuk address dan kontrol diatur oleh mikroprosesor 8088. Ketika suatu peralatan I/O hendak melakukan transfer data secara DMA maka ia mengaktifkan sinyal DMA Request. DMA Controller menerima sinyal tersebut dan mengirimkan sinyal Hold ke prosesor 8088 yang berarti bahwa DMA controller hendak mengambil alih pemakaian sistem bus. Pada akhir siklus, 8088 tidak lagi memegang kontrol atas sistem bus dan selanjutnya mengirim sinyal jawaban berupa Hold Acknowledge ke DMA Controller sebagai tanda bahwa sistem bus sudah bebas. Kemudian DMA Controller mengambil alih bus kontrol dan melakukan pemindahan data

[11]Liu, Yu-Cheng, opcit, hal. 395

antar peralatan I/O dan memory. Segera setelah DMA Controller menyelesaikan pemindahan datanya, prosesor 8088 mengambil alih kontrol atas sistem bus.

Saat menjadi master DMA Controller meletakkan address ke address bus dan mengirim sinyal yang diperlukan ke interface agar data tersedia di data bus. DMA Controller menentukan kapan sinyal bus request dimatikan, sehingga dapat mengembalikan kontrol sistem bus ke CPU segera setelah data ditransfer dan mengaktifkan sinyal bus request jika data lain telah tersedia, atau tetap memegang kontrol sampai seluruh blok data terkirim. Jalannya proses yang dilakukan dalam satu tindakan DMA adalah seperti terlihat pada gambar 2.12.

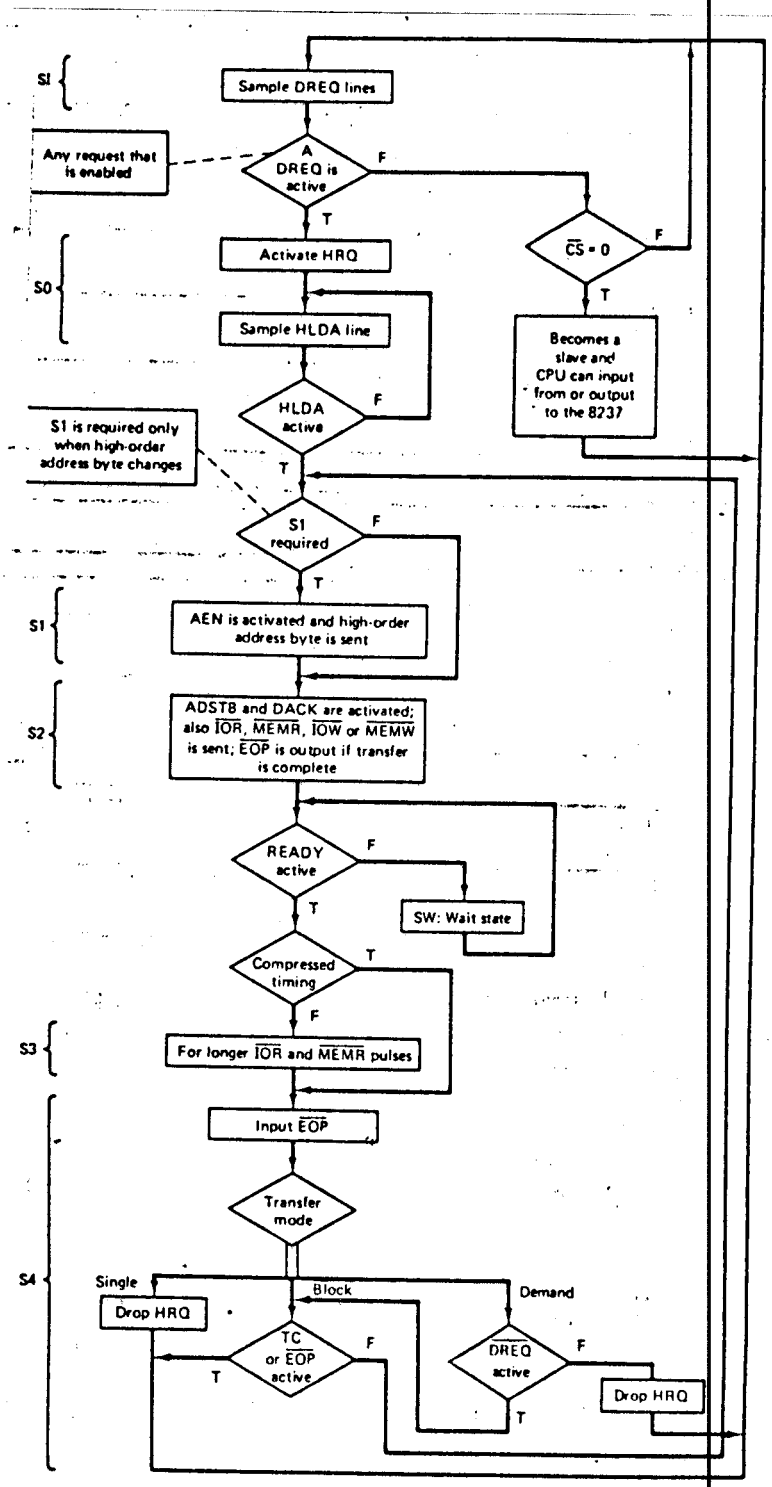
Adapun diagram waktu dari DMA Controller untuk suatu pemindahaan data dari/ke memory dapat dilihat pada gambar 2.13.

II.4 Filter Finite Impulse Response (FIR)

Jika dalam mendesain filter IIR (Infinite Impulse Rensponse) banyak digunakan metode transformasi dari yang telah dikembangkan dalam daerah waktu kontinyu, seperti filter Butterworth, Chebyshev dan lain-lain. Maka desain filter FIR sepenuhnya dikembangkan berdasar tehnik daerah waktu diskrit.

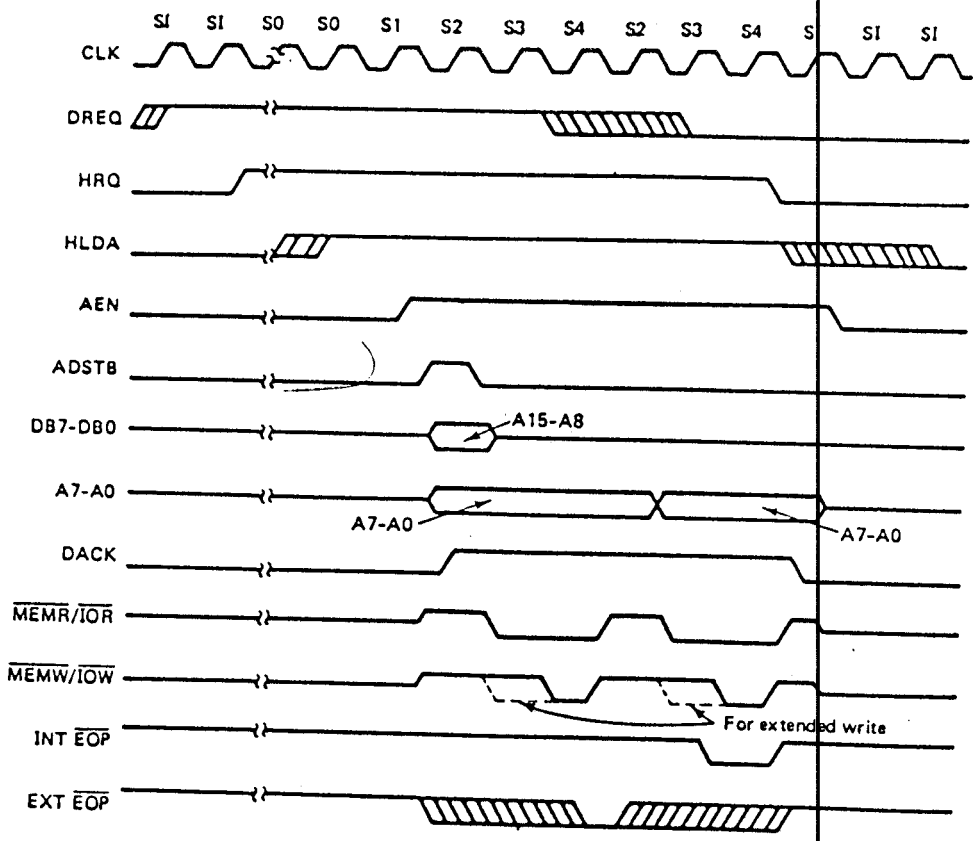
Kelebihan filter FIR dibanding filter IIR antara lain yaitu filter FIR mempunyai respon fase yang linier dan lebih mudah mendesain respon frekuensi yang beragam dibanding dengan metoda dalam filter IIR.

Metode termudah dalam mendesain suatu filter FIR adalah metode window. Metoda ini dimulai dengan penentuan daerah respon frekuensi yang



Gambar 2.12 Transfer data satu byte dalam suatu transfer blok data.[12]

[12]Ibid, hal. 403



Gambar 2.13 Diagram waktu untuk satu transfer dari DMA Controller

diinginkan dan ditulis seperti dibawah ini

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d[n]\ e^{-j\omega n} \qquad (2.3)$$

dimana $H_d[n]$[13] adalah impuls respons dari filter dan dapat ditulis dalam bentuk $H_d(e^{j\omega})$ sebagai

[13]Oppenheim, Alan V., *Discrete-Time Signal Processing*, Prentice Hall International Inc., hal. 440

$$h_d[n]=\frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})e^{j\omega n}d\omega \qquad (2.4)$$

Respon frekuensi dari filter ideal biasanya mempunyai diskontinyu pada daerah perbatasan antara pass band dan stop band seperti terlihat pada persamaan filter low pass dibawah ini:

$$H_d(e^{j\omega}) = \begin{cases} 1, & |\omega| \le \omega_c \\ 0, & \omega_c < |\omega| \le \pi \end{cases} \qquad (2.5)$$

Akibatnya impuls respons yang dihasilkan menjadi tidak berhingga dan non kausal. Cara yang langsung dan masuk akal untuk membuat filter FIR yang kausal adalah memotong impuls respons tersebut dengan suatu fungsi window. Fungsi window yang paling mudah adalah rectangular window yang didefinisikan seperti dibawah ini:

$$w[n] = \begin{cases} 1, & 0 \le n \le M, \\ 0, & \textit{lainnya.} \end{cases} \qquad (2.6)$$

Jadi impuls respons dari filter FIR yang baru adalah merupakan perkalian dari fungsi window *w[n]* dengan $h_d[n]$ sebagai berikut :

$$h[n] = h_d[n]w[n] \qquad (2.7)$$

sehingga diperoleh :

$$h[n] = \begin{cases} h_d[n], & 0 \le n \le M, \\ 0, & lainnya. \end{cases} \qquad (2.8)$$

Beberapa fungsi window lain yang biasa dipakai adalah terlihat pada gambar 2.14 dan didefinisikan seperti dibawah ini :

Hanning

$$w[n] = \begin{cases} 0.5 - 0.5\cos(2pin/M), & 0 \le n \le M, \\ 0, & lainnya. \end{cases} \qquad (2.9a)$$

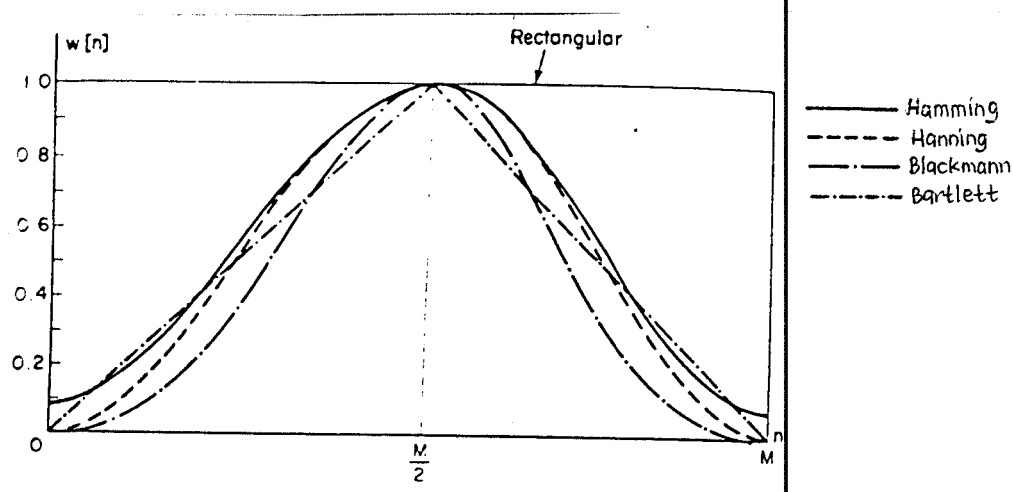
Hamming

$$w[n] = \begin{cases} 0.54 - 0.46\cos(2pin/M), & 0 \le n \le M, \\ 0, & lainnya. \end{cases} \qquad (2.9b)$$

Blackmann

$$w[n] = \begin{cases} 0.42 - 0.5\cos(2pin/M) + \cos(4pin/M), & 0 \le n \le M, \\ 0, & lainnya. \end{cases} \qquad (2.9c)$$

Perbandingan antara keempat fungsi window tersebut terlihat pada tabel 2.3. Seperti terlihat pada tabel 2.3 ada pertentangan antara lebar main lobe dengan ripple ratio, yaitu semakin kecil ripple ratio dari window ke window makin lebar mainlobe-nya. Parameter terakhir ini dapat diatur dengan mengubah order filter (N). Sebaliknya, ripple ratio rata-rata konstan untuk suatu window tertentu. Jadi untuk memperoleh spesifikasi filter yang diinginkan harus dipilih dulu window



Gambar 2.14 Beberapa window yang biasa digunakan[14]

Tabel 2.3 Perbandingan antara beberapa window yang biasa digunakan[15]

Model window	lebar mainlobe	Ripple ratio, %			
		N = 11	N = 21	N	= 31
Recatangular	$2\omega_s/N$	22.34	21.89		21.8
Hanning	$4\omega_s/N$	2.62	2.67		2.67
Hamming	$4\omega_s/N$	1.47	0.93		0.82
Blackman	$6\omega_s/N$	0.08	0.12		0.12

yang mempunyai ripple ratio yang sesuai dan kemudian memilih orde yang sesuai untuk memperoleh lebar daerah transisi yang cocok. Karena jumlah ripple ratio terbatas sejumlah window yang tersedia maka desain filter biasanya memilih window yang mempunyai ripple ratio terlalu kecil yang akan menghasilkan lebar mainlobe tinggi yang berlebih. Akibatnya untuk mengecilkan daerah transisi maka variable N dinaikkan yang akan meningkatkan orde dari filter secara berlebih.

---

[14]Ibid, hal. 447

[15]Antoniou, Andreas, ***Digital Filters : Analysis and Design***, Tata McGraw-Hill Co. Ltd., hal. 234

Untuk memecahkan masalah tersebut diatas ditemukan suatu window oleh Kaiser yang dapat mengubah ripple ratio dari yang terkecil (window Blackman) sampai yang terbesar (window Rectangular) disamping mengubah lebar mainlobe seperti window yang terdahulu.

Window Kaiser didefinisikan sebagai berikut:

$$w[n] = \begin{cases} \dfrac{I_o(\beta)}{I_o(\alpha)}, & |n| \le \dfrac{N-1}{2}, \\ 0, & \textit{lainnya.} \end{cases} \qquad (2.10)$$

dimana α adalah parameter independen dan

$$\beta = \alpha\sqrt{1-\left(\frac{2n}{N-1}\right)^2} \qquad (2.11)$$

$I_o(x)$ adalah fungsi Bessel orde ke nol dan didefinisikan sebagai:

$$I_o(x) = 1 + \sum_{k=1}^{\infty}\left[\frac{1}{k!}\left(\frac{x}{2}\right)^k\right]^2 \qquad (2.12)$$

Passband ripple dan pelemahan (*attenuation*) stopband dalam desibel didefinisikan sebagai :

$$A_p = 20 \log \frac{1+\delta}{1-\delta} \qquad (2.13)$$

$$A_a = -20 \log \delta \qquad (2.14)$$

Filter bandstop dengan respon frekuensi ideal seperti terlihat pada gambar 2.15 mempunyai passband ripple sama atau lebih kecil dari A_p, pelemahan stopband sama atau lebih besar dari A_a dan lebar daerah transisi B_t dapat didesain dengan prosedur sebagai berikut :

1. Dapatkan *h[n]* dari respon frekuensi ideal filter yaitu :

$$H(e^{j\omega}) = \begin{cases} 1, & \text{untuk } 0 \leq |\omega| \leq \omega_{c1} \\ 0, & \text{untuk } \omega_{c1} < |\omega| < \omega_{c2} \\ 1, & \text{untuk } \omega_{c2} \leq |\omega| < \pi \end{cases} \qquad (2.15)$$

dimana

$$\omega_{c1} = \omega_{p1} + \frac{B_t}{2} \qquad \omega_{c2} = \omega_{p2} - \frac{B_t}{2} \qquad (2.16)$$

2. Ambil δ dalam persamaan (2.13) dan (2.14) sehingga $A_p \leq A'_p$ dan $A_a \geq A'_a$. Harga yang cocok dapat diambil menggunakan rumus dibawah ini:

$$\delta = \min(\delta_1, \delta_2) \qquad (2.17)$$

dimana

$$\delta_1 = 10^{-0.05A_a'} \qquad \delta_2 = \frac{10^{0.05A_p'} - 1}{10^{0.05A_p'} + 1} \qquad (2.18)$$

3. Hitung harga A_a menggunakan persamaan (2.14).

4. Pilih parameter α sebagai berikut :

$$\alpha = \begin{cases} 0 & A_a \leq 21 \\ 0.5842(A_a - 21)^{0.4} + 0.07886(A_a - 21) & 21 < A_a \leq 50 \\ 0.1102(A_a - 8.7) & A_a > 50 \end{cases} \qquad (2.19)$$

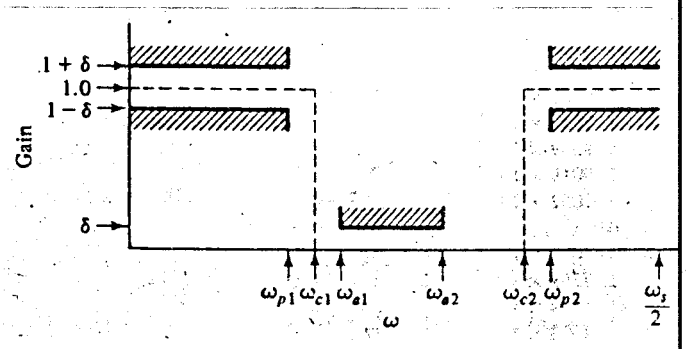
5. Pilih parameter *D* sebagai berikut :

$$D = \begin{cases} 0.9222 & \text{untuk } A_a \leq 21 \\ \dfrac{A_a - 7.95}{14.36} & \text{untuk } A_a > 21 \end{cases} \qquad (2.20)$$

Selanjutnya diambil bilangan ganjil N terkecil yang memenuhi pertidaksamaan

$$N \geq \frac{\omega_{sD}}{B_t} + 1 \qquad (2.21)$$

6. Mencari persamaan *w[n]* menggunakan persamaan (2.10).



Gambar 2.15 Respon frekuensi ideal dari filter bandstop

II.5 Konversi Analog ke Digital

Sistem mikroprosesor hanya dapat mengolah data dalam bentuk biner saja atau lebih sering disebut besaran digital, oleh sebab itu setiap data analog yang akan diproses oleh mikroprosesor harus diubah lebih dulu ke dalam bentuk biner.

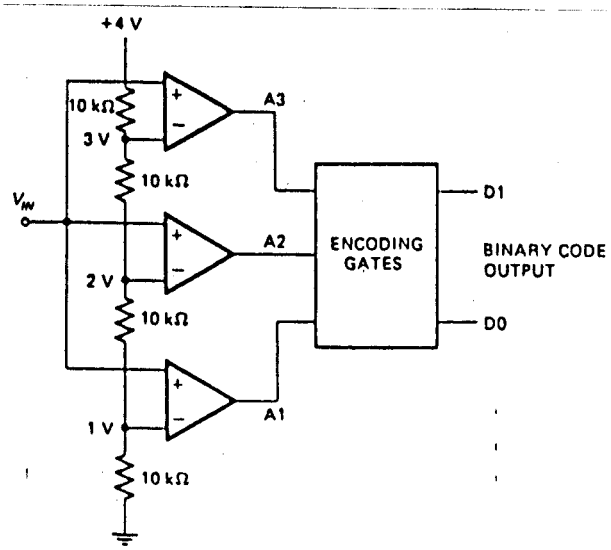
Banyak IC dewasa ini yang dapat langsung mengubah besaran analog menjadi digital, masing-masing memiliki karakteristik dan keunggulan sendiri-sendiri. Karena dalam tugas akhir ini dibutuhkan pengubahan data analog menjadi

digital dengan kecepatan tinggi, maka ADC yang diuraikan di sini dibatasi untuk yang memiliki kecepatan tinggi (flash).

FLASH ADC

Konsep yang paling sederhana dan paling cepat dari berbagai jenis ADC adalah *Paralel Comparator* atau yang biasa disebut *Flash ADC*. Gambar 2.16 memberikan skema sederhana untuk ADC jenis ini. Hubungan antara A1, A2 dan A3 dengan D0 dan D1 dapat dilihat pada tabel 2.4.

Dalam contoh di atas terlihat bahwa untuk 2 bit ADC, dibutuhkan (4-1) komparator. Ini berarti untuk n bit Flash ADC harus digunakan (2^n-1) komparator. Inilah kerugian menggunakan *Paralel Comparator* yaitu jumlah komparator yang digunakan banyak.



Gambar 2.16 Paralel Comparator ADC[16]

[16]Hall, Douglas V., op. cit., 1983.

Tabel 2.4 Hubungan output binary dengan output komparator[17]

Vin	Output Comparator			Binary Output	
	A1	A2	A3	D1	D0
0 - 1/4 VRef	0	0	0	0	0
1/4 - 2/4 VRef	1	0	0	0	1
2/4 - 3/4 VRef	1	1	0	1	0
3/4 - VRef	1	1	1	1	1

[17]Ibid, hal. 167

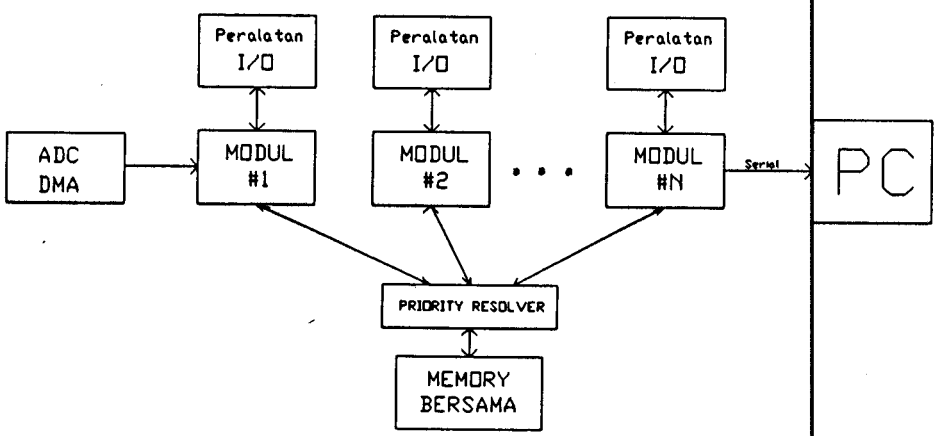
BAB III

PERENCANAAN PERANGKAT KERAS

III.1 Blok Umum Perencanaan Perangkat Keras

Dalam Tugas Akhir ini akan direncanakan suatu sistem modul 8088 mode maksimum paralel yang bersifat universal sehingga dapat digunakan pada keperluan lain yang membutuhkan kecepatan dalam prosesnya. Pengambilan data sinyal spike (*impuls*) merupakan proses yang membutuhkan kecepatan tinggi.

Keseluruhan peralatan dibagi menjadi beberapa bagian, yaitu modul paralel 8088, modul ADC, modul modul interface memory bersama dan modul serial yang menghubungkan sistem dengan monitor yang dalam hal ini berupa komputer PC.



Gambar 3.1 Diagram blok secara umum

Gambar 3.1 menunjukkan diagram blok secara umum dari sistem yang direncanakan, yaitu peralatan bekerja dalam mode *stand alone* yang tidak

bergantung pada suatu induk prosesor (*host*). Gambar 3.1 juga menunjukkan ada proses kerja paralel dari sejumlah modul, yaitu ada modul yang berfungsi sebagai pengambil data dan ada modul yang berfungsi mengirim data ke PC.

Data yang diambil oleh modul pertama selanjutnya secara software diproses oleh modul pertama bersama-sama (*concurently*) dengan modul lain dan kemudian hasil akhirnya dikirim oleh modul terakhir ke komputer. Komunikasi antar modul prosesor tidak dapat dilakukan secara langsung dan sebagai gantinya komunikasi antar modul paralel 8088 dilakukan melalui memori bersama dimana prioritas pemakaiannya ditentukan oleh sebuah kontrol prioritas (*priority resolver*).

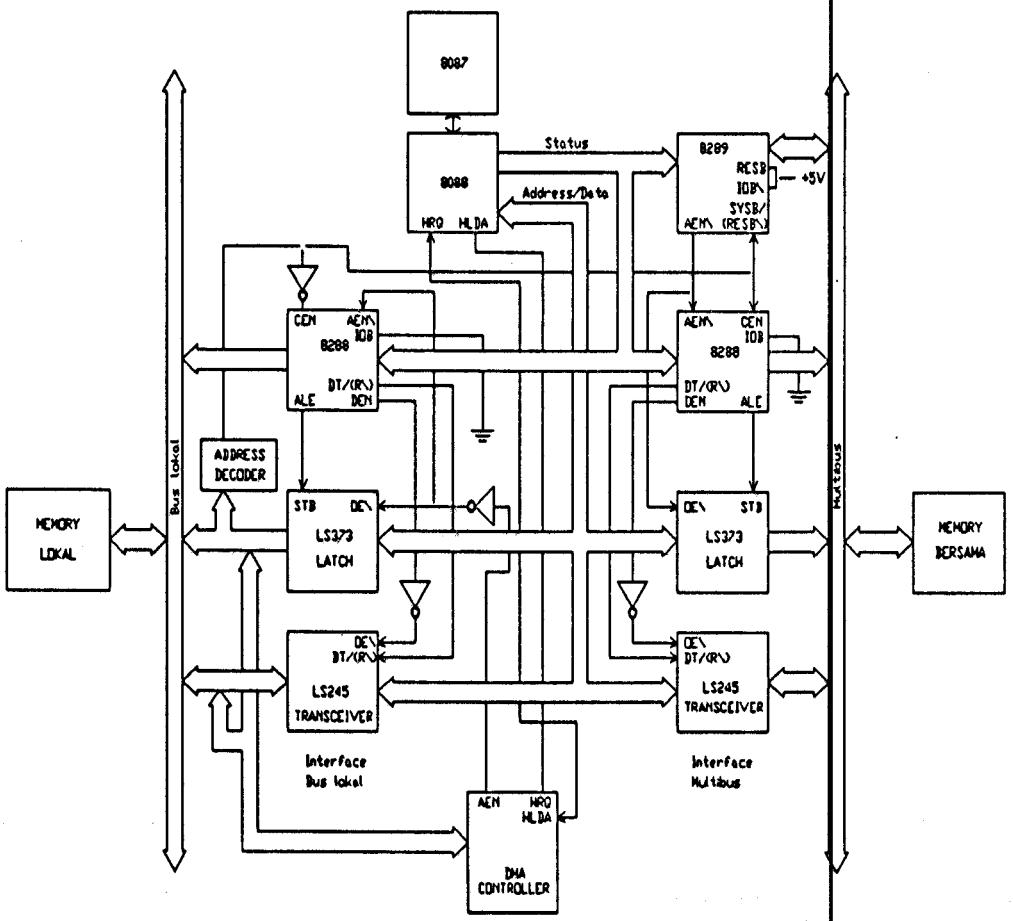
Dalam perencanaan ini modul yang dibuat adalah sebanyak tiga buah modul. Hal ini dengan pertimbangan proses pengambilan event spike dapat dibagi menjadi tiga bagian yaitu proses pengambilan data, proses FFT, proses IFFT dan pengecekan data.

Tiap modul paralel disamping dapat mengakses memory bersama juga mempunyai memory lokal yang diperuntukkan khusus bagi modul itu saja. Penyediaan memory lokal ini dimaksudkan untuk mengurangi aktifitas pemakaian dari multibus, karena jika pemakaian multibus mencapai titik jenuh maka kecepatan proses paralel tidak dapat ditingkatkan lagi walau modul prosesor ditambah.

Dalam pengembangan lebih lanjut posisi komputer PC diatas dapat digantikan dengan media printer atau disk drive sebagai penyimpan data sehingga proses secara keseluruhan dapat lebih cepat lagi karena penggunaan komunikasi serial dapat dihindari.

III.2 Modul paralel 8088

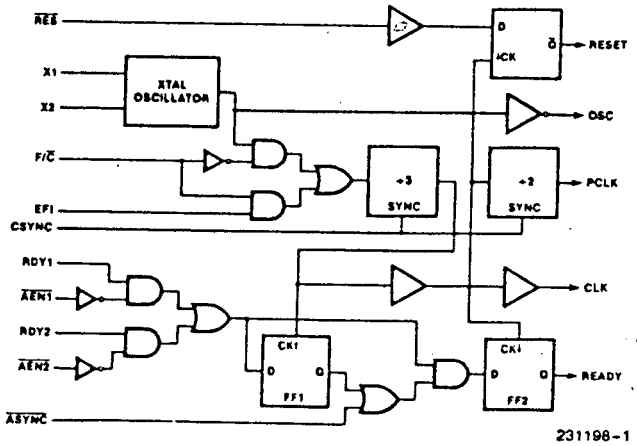
Secara garis besar sistem modul 8088 paralel terbagi atas beberapa bagian, yaitu pembangkit clock 8288 yang berfungsi menyuplai clock dan mengeluarkan sinyal READY bagi prosesor, bus controller 8288 yang bertugas mengaktifkan sinyal kontrol bus baik bus lokal maupun multibus, prosesor 8088 sendiri, co-prosesor 8087 untuk kalkulasi bilangan real, DMA Controller dan Bus Arbiter 8289 yang berfungsi untuk mengakses multibus. Diagram blok yang menunjukkan hubungan antara bagian-bagian itu semua dapat dilihat pada gambar 3.2.



Gambar 3.2 Blok Diagram Modul 8088

III.2 Sinkronisasi Sistem Multimaster oleh 8284

Bagian penting dari 8284 yang tidak ditemui pada pembangkit clock lainnya adalah kemampuan sinkronisasi pada sistem multimaster melalui output READY. Output ini dipengaruhi oleh dua pasang input AENx\ dan RDYx seperti terlihat pada gambar 3.3. Tanda x disini menunjukkan bilangan indeks dan juga untuk membedakan dengan sinyal AEN\ yang berasal dari DMA Controller.



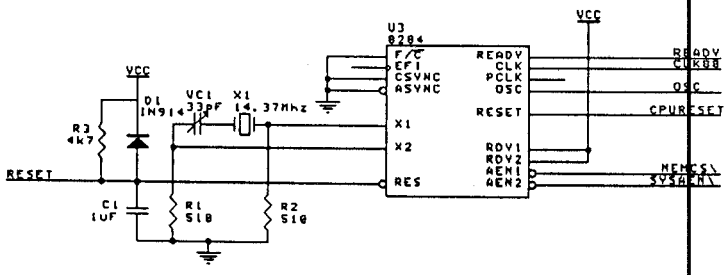
Gambar 3.3 Blok diagram pembangkit clock 8284[18]

Agar sepasang input AENx\ dan RDYx dapat mempengaruhi output READY maka sepasang input lainnya harus dibuat non aktif.

Dari perencanaan ada dua bagian yang membutuhkan penundaan proses dari CPU yaitu pada proses transfer DMA dan pada proses pemakaian multibus. Hal ini diimplementasikan dengan cara input AEN1\ diberi sinyal MEMCS\ yang merupakan keluaran dari address dekoder memory lokal dan AEN2\ diberi input sinyal SYSAEN\ yang berasal dari bus arbiter 8289. Rangkaiannya adalah seperti pada gambar 3.4.

---

[18] _____, *Microsystem Components Handbook : Microprocessor Volume I*, Intel Corporation 1986, hal. 3-246



Gambar 3.4 Logic input RDY ke 8284

Sinyal SYSAEN\ jika aktif (low) maka berarti multibus sudah bisa digunakan. Sedangkan MEMCS\ jika aktif berarti prosesor mengakses memory lokal. Dengan konfigurasi ini jika prosesor mengakses lokal memory maka MEMCS\ low dan SYSAEN\ high (non aktif) sehingga output READY akan high karena AEN1 low. Saat prosesor mengakses memory bersama maka MEMCS\ akan high tetapi SYSAEN\ tidak langsung aktif (low) karena masih menunggu hak pemakaian terhadap multibus. Akibatnya ada waktu dimana kedua input AEN dalam keadaan high yang berarti output READY akan menjadi low (*not ready*) yang berarti prosesor akan memasuki keadaan tunggu (*wait state*) sampai arbiter memperoleh hak pemakaian atas multibus. Setelah arbiter mendapat hak kontrol atas multibus SYSAEN\ akan berubah menjadi low. Akibatnya READY kembali high dan prosesor kembali mengerjakan cycle yang tertunda.

Input READY tidak disinkronisasi oleh 8088 sehingga diperlukan komponen tambahan untuk keperluan tersebut. Output READY 8284 sudah merupakan sinyal yang sinkron yaitu terhadap transisi turun dari clock dan memenuhi persyaratan waktu hold dan set sinyal READY.

Sinyal READY digunakan oleh 8088 untuk mengecek apakah peralatan

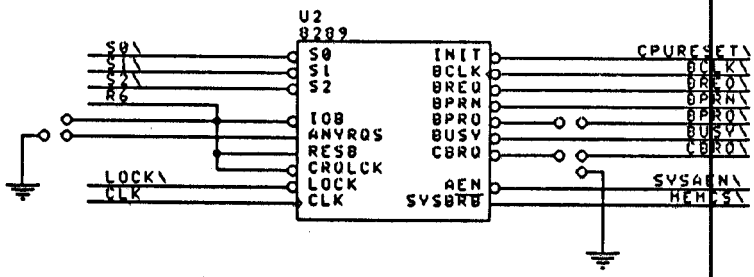
pendukung lain sudah menyelesaikan tugasnya. Jika belum maka 8088 akan menambah waktu tunggu (Tw) antara T3 dan T4. 8088 mengecek keberadaan READY pada akhir T2, T3 dan Tw untuk menentukan apakah perlu menambah Tw lagi. Sehingga agar 8088 bisa masuk ke keadaan tunggu (wait state) maka sinyal READY harus low (not ready) pada waktu-waktu diatas.

III.2.2 Bus Arbiter 8289

Dalam perencanaan sistem ini bus arbiter diset pada mode bus residen, yaitu pin IOB dan RESB diset pada logic satu. Pemakaian atas bus lokal atau multi bus ditentukan oleh input ke pin SYSB/(RESB\). Jika input pada pin ini low maka prosesor memberitahu arbiter jika sedang mengakses lokal bus. Jika input high berarti prosesor hendak mengakses sistem multibus dan disini baru terlihat fungsi dari bus arbiter ini, yaitu arbiter akan mengeluarkan sinyal BREQ\ yang berarti prosesor hendak menggunakan multibus dan selanjutnya output sinyal ini dimasukkan ke priority resolver dari memory bersama. Jika pada saat itu prioritas arbiter ini paling tinggi dalam permintaan multibus maka priority resolver akan menjawab berupa sinyal low ke input BPRN\ arbiter yang meminta pemakaian multibus tersebut. Setelah mendapat jawaban low pada pin BPRN\, arbiter mengeluarkan sinyal kontrol SYSAEN\ yang berarti hak pemakaian multibus sudah diperoleh dan prosesor bisa segera mengakses memori bersama. Sinyal SYSAEN\ ini dalam perencanaan digunakan untuk mengaktifkan buffer address dan kontroler multibus.

Input lainnya yang diatur adalah pin ANYRQST (diset high) yang berarti

arbiter menerima permintaan pemakaian atas multibus dan akan melepas pemakaian multibus walaupun permintaan berasal dari arbiter dengan prioritas lebih rendah. Pin CBRQ\ dihubungkan dengan pin CBRQ\ arbiter dari modul lainnya. Pin ini jika low menandakan ada modul lain akan menggunakan multibus. Input CRQLCK\ diset high yang berarti arbiter dapat melepas pemakaian multibus ke arbiter yang lebih rendah prioritasnya. Rangkaian dari bus arbiter 8289 ini terlihat pada gambar 3.5.

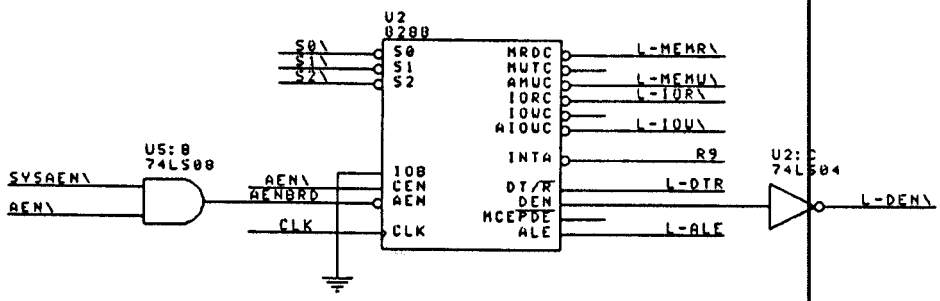


Gambar 3.5 Bus arbiter 8289

III.2.3 Pengontrolan Bus Controller 8288

Ada dua input yang mempengaruhi bekerjanya bus controller 8288 dalam mode bus sistem, yaitu Command Enable (CEN) dan Address Enable (AEN\). AEN\ berfungsi untuk meng-enable keluaran perintah dan CEN mempunyai tugas mengaktifkan keluaran perintah. CEN dapat digunakan sebagai pemisah daerah pemakaian memory, yaitu jika suatu alamat memory berada pada bus lokal maka perintah yang dikeluarkan adalah berasal dari kontroler bus lokal bukan dari kontroler multi bus dengan cara mematikan input CEN kontroler multibus. Sebaliknya jika lokasi memory berada pada multi bus maka perintah dikeluarkan kontroler multi bus dan CEN kontroler bus lokal dinon-aktifkan.

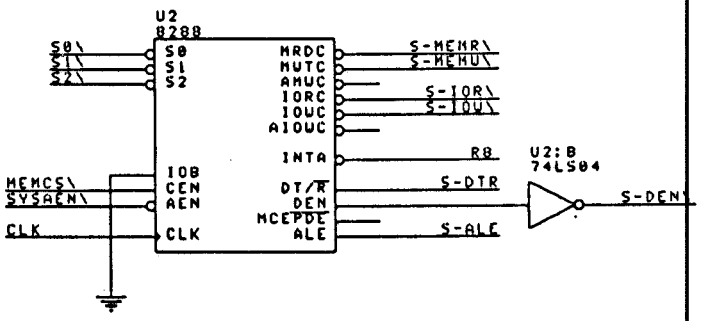
Pada lokal bus terdapat sebuah DMA Controller yang mempunyai potensi sebagai master atas lokal bus, yaitu saat operasi DMA. Oleh karena itu input CEN diset sehingga aktif hanya saat mengakses memory lokal dan tidak sedang melakukan proses DMA, yaitu sinyal SYSAEN\ high dan sinyal AEN\ (yang menandakan proses transfer DMA sedang berlangsung) juga high. AEN\ jika high berarti sedang terjadi proses DMA dan jika low berarti tidak ada proses DMA. Sedangkan untuk input AEN dari 8288 diset oleh sinyal AENBRD. Sinyal AENBRD adalah sinyal kebalikan dari AEN\. Jadi jika terjadi proses DMA input AEN 8288 akan low dan keluaran dari kontroler bus lokal akan berada dalam keadaan *high impedance*. Rangkaian konfigurasi untuk kontroler bus lokal ini dapat dilihat pada gambar 3.6.



Gambar 3.6 Kontroler untuk bus lokal

Sedangkan untuk kontroler multibus input CEN dihubungkan ke MEMCS\ dan input AEN dihubungkan dengan sinyal SYSAEN\. Dengan keadaan seperti ini maka dipastikan output perintah kontroler multibus aktif saat prosesor mengakses sistem multibus dan ini juga berarti saat yang sama kontroler bus lokal dalam keadaan non aktif, karena input CEN-nya juga dipengaruhi oleh sinyal

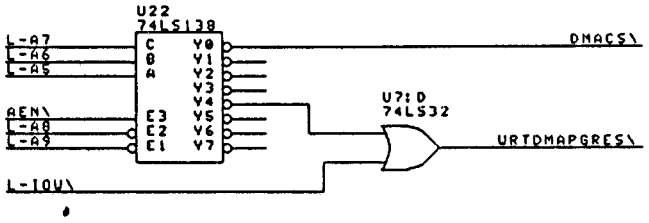
SYSAEN\. Rangkaian kontroler multibus dapat dilihat pada gambar 3.7.



Gambar 3.7 Kontroler untuk multibus

III.2.4 DMA Controller

DMA Controller dalam keadaan tanpa kegiatan (idle) dapat diprogram sebagai I/O dengan address 00 sampai 0FH. dan rangkaiannya adalah seperti pada gambar 3.8.

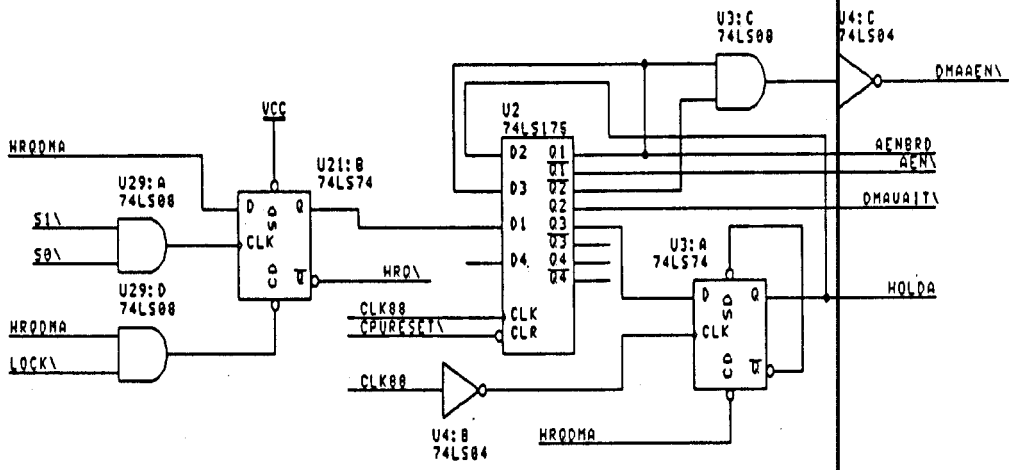


Gambar 3.8 Dekoder untuk DMA controller dan page register

Permintaan HOLD ke CPU oleh DMA

Pada mode maksimum 8088 tidak mempunyai fasilitas sinyal HOLD dan HLDA seperti mode minimum, dan sebagai gantinya adalah pin RQ/GT yang merupakan multiplex dari sinyal input request (RQ\) dan sinyal output grant (GT\). DMA controller tidak mempunyai sinyal RQ/GT melainkan sinyal HOLD dan HLDA. Karena perbedaan ini maka dibuat suatu rangkaian logic yang mena-

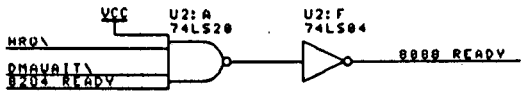
ngani sinyal dari DMA ini tanpa menggunakan sinyal RQ/GT dari 8088. Hal ini dilakukan dengan mengecek sinyal status S1, S0 dan LOCK. Sinyal S1 dan S0 jika high maka CPU dalam keadaan passive atau halt dan LOCK jika high maka CPU tidak sedang melaksanakan instruksi yang mempunyai awalan LOCK. Sinyal LOCK ini merupakan output CPU yang berarti saat melaksanakan instruksi tersebut CPU tidak bisa diganggu oleh proses lain baik dari DMA, 8087 maupun permintaan pemakaian multi bus. Rangkaiannya adalah seperti pada gambar 3.9.



Gambar 3.9 Rangkaian pengaktif sinyal HLDA bagi DMA

Sinyal HRQDMA adalah sinyal yang asinkron, sehingga agar persyaratan pengenalan sinyal not ready pada 8088 terpenuhi maka dirancang jawaban HOLDA akan high jika permintaan HRQDMA timbul sebelum status 8088 high pada T3. Atau dengan kata lain HRQDMA disinkronkan terhadap transisi naik dari status. Transisi naik dari status ini terjadi pada state T3 dan jika keluaran not ready ini dimasukkan ke input RDY 8284 maka output READY 8284 akan low pada T4 yang berarti menyalahi persyaratan waktu sinyal input READY 8088.

Untuk itu output not ready keluaran dari DMA dihubungkan langsung dengan input READY. Gambar 3.10 menunjukkan hubungan dari sinyal not ready dari DMA dan READY 8284.



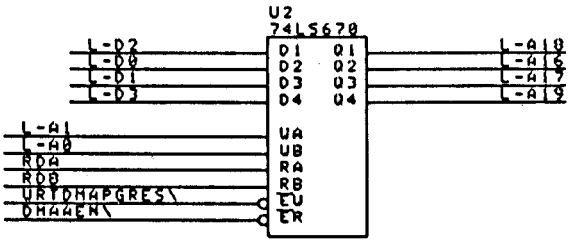
Gambar 3.10 Input READY 8088

Jawaban sinyal HLDA dibuat secepat mungkin, setelah ada HOLD request dan semua kondisi terpenuhi, yaitu setengah clock CPU untuk menghidari perubahan status CPU. Untuk itu input clock D flip-flop diberi masukan clock CPU yang dibalik. Jadi jika saat transisi naik dari CLK sinyal HOLD high (aktif) maka pada saat transisi turun CLK sinyal HLDA akan high (aktif) pula.

Selanjutnya pada setengah clock berikutnya sinyal HLDA mengaktifkan sinyal kontrol AENBRD yang berfungsi untuk mematikan bus controller, address latch, address buffer dan data transceiver sehingga saat proses DMA berlangsung hanya sinyal address dari DMA yang aktif. Sinyal AENBRD ini kemudian mengaktifkan sinyal DMAWAIT\ yang berfungsi memberitahu CPU kalau proses DMA sedang berlangsung sehingga CPU memasuki keadaan tunggu (wait state).

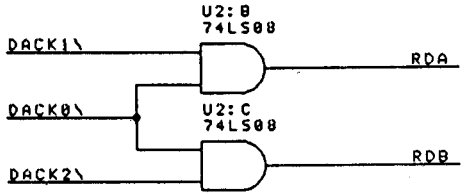
DMA controller dalam operasinya hanya bisa mengirim satu blok data 64 kilo byte karena address yang dikeluarkannya hanya 16 bit. Untuk dapat mencapai semua lokasi memori maka ditambah suatu page register 4 bit untuk 4 bit address tertinggi. Page register ini diakses sebagai I/O port dengan alamat berurutan 80H sampai 83H untuk channel 0 sampai 3. Rangkaian page register ini dapat dilihat

pada gambar 3.11.



Gambar 3.11 Page register untuk DMA Controller

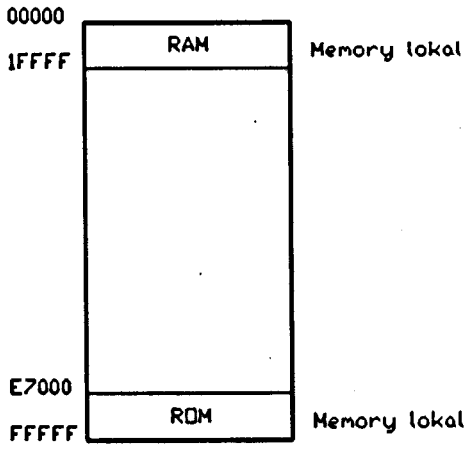
Untuk mengeluarkan 4 bit address maka page register harus dalam mode baca. Enable baca dari page register ini diaktifkan oleh sinyal DMAAEN\. Sinyal DMAAEN\ ini baru aktif satu clock kemudian setelah AENBRD aktif, bersamaan dengan aktifnya sinyal AENBRD. Sinyal DMAAEN\ ini disamping itu digunakan juga untuk mengaktifkan address latch DMA. Selanjutnya proses pengeluaran 4 bit address dilakukan dengan memakai sinyal DACK dari masing-masing channel. Karena input baca pada page register membutuhkan bilangan binary biasa untuk menentukan register mana yang akan mengeluarkan address maka dilakukan pengkodean sehingga jika DACK0\ aktif maka register 0 akan mengeluarkan address, DACK1\ mengaktifkan register 1 dan seterusnya. Pengkodean sinyal DACK\ adalah seperti pada gambar 3.12.



Gambar 3.12 Pengkodean sinyal DACK\

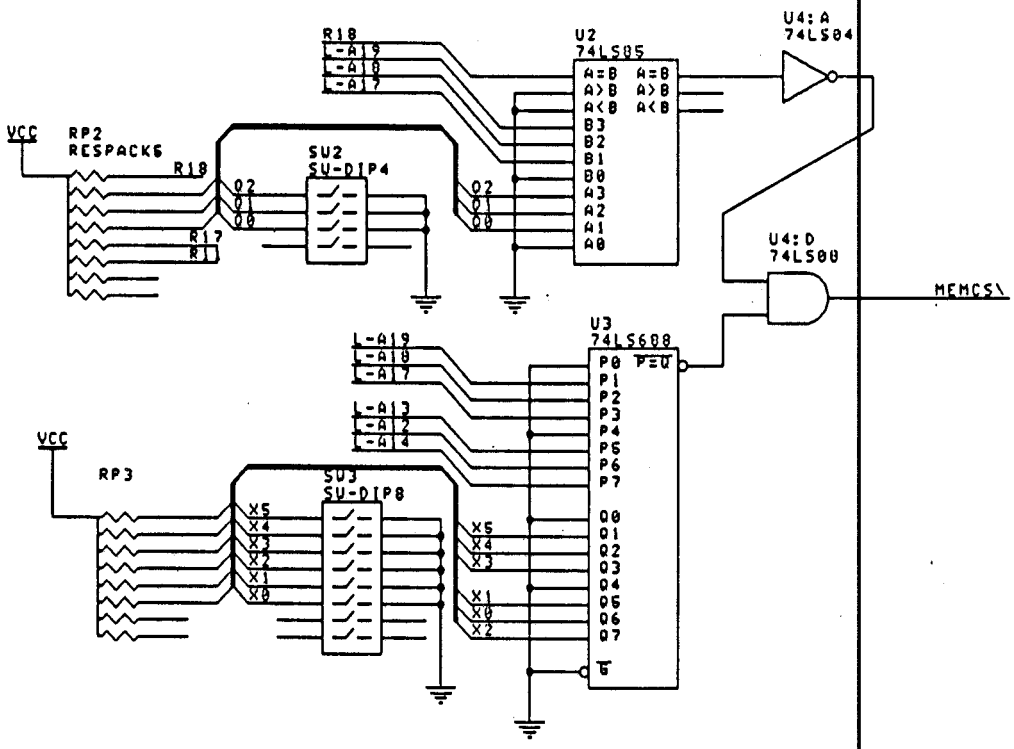
III.2.5 Decoder memory lokal dan memory bersama

Dalam perencanaan ini dipilih dua buah memori lokal yaitu 4K ROM dengan lokasi dari FF000H sampai FFFFFH dan 32K RAM statis dengan lokasi alamat dari 00000H sampai 07FFFH. Dan selanjutnya alamat diluar lokasi tersebut diatas direncanakan berada di memori bersama. Map memory dari sistem yang direncanakan adalah seperti pada gambar 3.13.



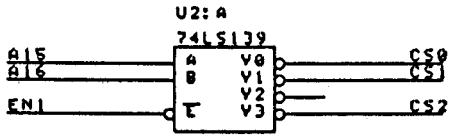
Gambar 3.13 Map memory lokal dan memory bersama

EPROM 4 kilo mempunyai 12 bit alamat, sehingga 8 bit sisanya digunakan untuk decoder dan RAM statis 32 kilo mempunyai 15 bit alamat dan 5 bit sisanya digunakan sebagai dekoder enable CS.



Gambar 3.14 Dekoder memory lokal

Gambar 3.14 menunjukkan rangkaian dekoder dari memori lokal. Dari rangkaian dekoder dan perencanaan map memory terlihat MEMCS\ akan low jika address A17 sampai A19 sama dengan harga 000 atau 111. Jika A17 sampai A19 diluar harga tersebut maka MEMCS\ akan high yang berarti address berada di memory bersama. Address A15 dan A16 tidak diikutkan dalam proses penentuan aktif tidaknya MEMCS\ namum digunakan sebagai selector antara RAM dan ROM di modul terpisah, yaitu modul memory lokal. Untuk lebih jelasnya dapat dilihat pada gambar 3.15.



Gambar 3.15 Selector antara RAM dan ROM memory lokal

Dengan konfigurasi address dan dekoder seperti diatas maka kemungkinan overlap alamat dari RAM dan ROM adalah seperti terlihat pada tabel 3.1 dan 3.2.

Tabel 3.1 Alamat ROM lokal

A19 18 17 16 . 15 14 13 12 ¦ 11 ... 0	KETERANGAN
1 1 1 0 . 0 1 1 1 ¦ x ... x	E7000H - E7FFFH *
1 1 1 0 . 1 1 1 1 ¦ x ... x	EF000H - EFFFFH
1 1 1 1 . 0 1 1 1 ¦ x ... x	F7000H - F7FFFH
1 1 1 1 . 1 1 1 1 ¦ x ... x	FF000H - FFFFFH **

Tabel 3.2 Alamat RAM lokal

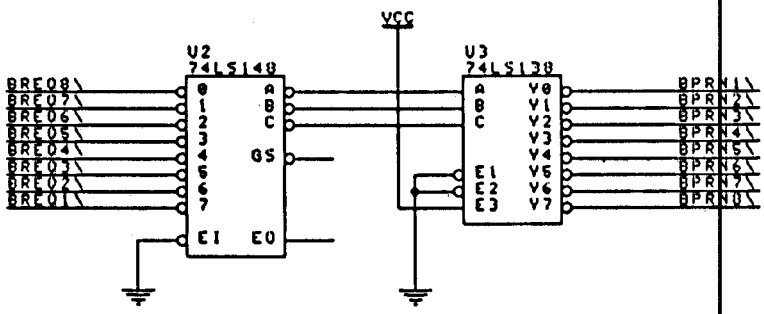
A19 18 17 16 . 15 ¦ 14 ... 0	KETERANGAN
0 0 0 0 . 0 ¦ x ... x	00000H - 07FFFH *
0 0 0 0 . 1 ¦ x ... x	08000H - 0FFFFH
0 0 0 1 . 0 ¦ x ... x	10000H - 17FFFH
0 0 0 1 . 1 ¦ x ... x	18000H - 1FFFFH **

Alamat 00000 - 07FFF overlap dengan E7000 - 0E7FFF dan alamat FF000 - FFFFF overlap dengan 18000 - 1FFFF seperti terlihat pada bagian yang bertanda bintang. Lokasi diluar daerah memory diatas adalah termasuk memory bersama.

III.3 Modul Interface Memori Bersama

1. Priority resolver pada modul memory bersama

Rangkaian lengkapnya adalah seperti pada gambar 3.16.

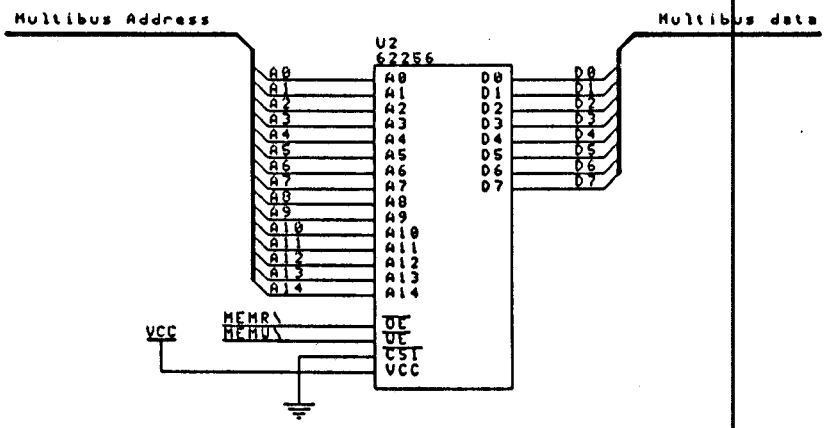


Gambar 3.16 Rangkaian priority resolver

Priority resolver ini terdiri dari sepasang decoder encoder. Input request bus dimasukkan pada encoder yang akan diubah menjadi bilngan binary. Selanjutnya bit ini didecodekan menjadi keluaran prioritas mana yang berhak menggunakan multibus. Enable pada kedua komponen dihubungkan dengan ground yang berarti priority resolver ini selalu aktif.

2. Pemakaian alamat dari memory bersama

Alamat A15 sampai A19 dari memory bersama tidak digunakan sebagai enable chip select, jadi dengan rancangan ini semua alamat yang tidak membuat MEMCS\ aktif (low) maka address tersebut akan mengakses memory bersama. Enable dari memori bersama ini diset pada logic satu sehingga selalu siap menerima akses dari modul mana saja yang menggunakan multibus. Rangkaiannya adalah seperti terlihat pada gambar 3.17.



Gambar 3.17 Memory bersama dari modul-modul 8088

III.4 Modul Analog to Digital Converter (ADC)

ADC yang digunakan disini adalah MC10319 dari Motorola. ADC tersebut merupakan flash ADC yang mempunyai spesifikasi sebagai berikut:

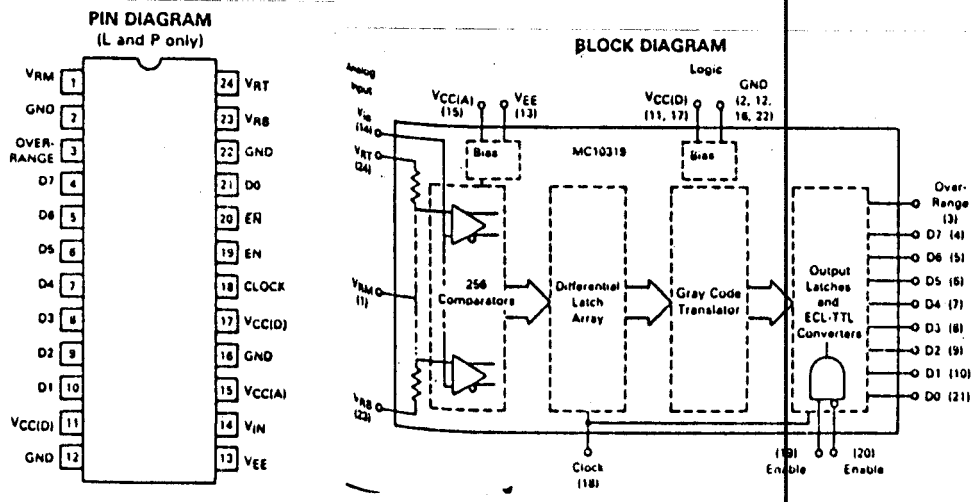
- sampling rate 25 MHz
- resolusi 8 bit
- input range 1.0 - 2.0 Vpp
- konversi 1 clock cycle
- output 3 state LS TTL

Diagram pin dan diagram blok diperlihatkan seperti pada gambar 3.18.

Power Supply

ADC ini mempunyai dua buah bagian power supply. VCC(a) (pin 15) adalah positip power supply untuk komparator (bagian analog) dan VCC(b) (pin 11,17) adalah positip power supply untuk bagian digital. Keduanya diberikan tegangan +5 volt. VEE adalah negatip power supply untuk komparator. Tegang-

annya berkisar antara -3 sampai -6 volt dan paling tidak 1.3 volt lebih negatip dari VBB.



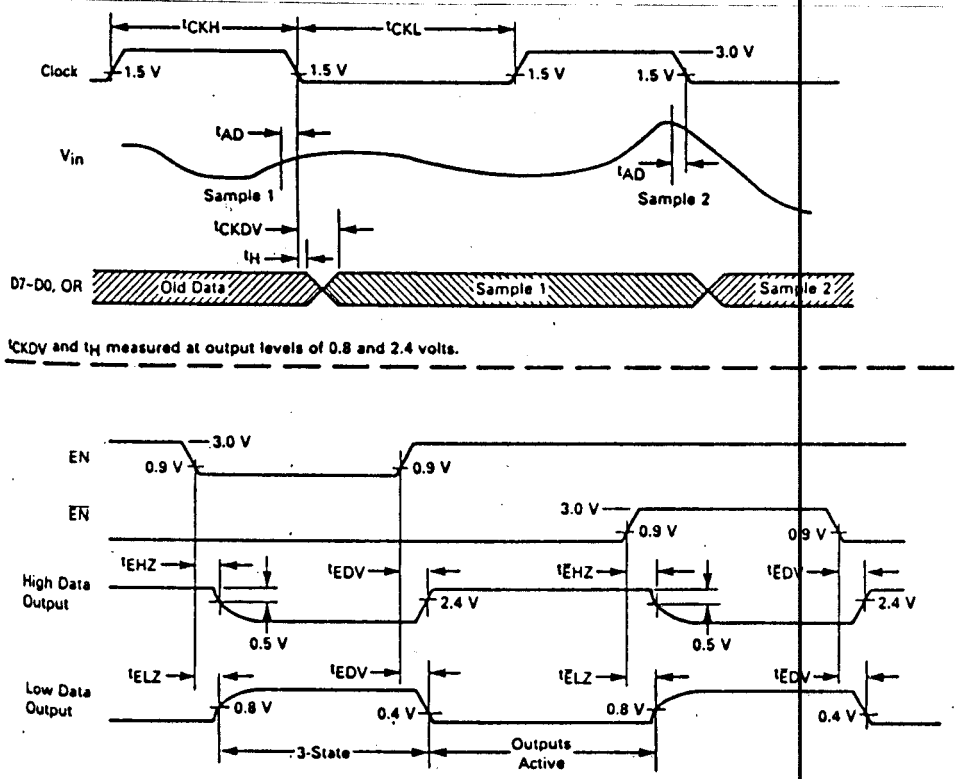
Gambar 3.18 Diagram pin dan diagram blok MC10319

Enable Input

Dua buah input enable (pin 19, 20) kompatibel dengan TTL dan digunakan untuk merubah data output (D7-D0) dari kondisi aktif ke kondisi 3-state. ADC ini direncanakan selalu dalam keadaan siap konversi sehingga input kontrol enable (pin 19 dan 20) dibuat aktif, yaitu pin 19 diberi logic high dan pin 20 diberi logic low.

Clock

Input clock (pin 18) kompatibel dengan TTL, mempunyai range frekuensi 0 sampai 30 MHz tanpa batasan duty cycle. Timing diagram ADC ini ditunjukkan pada gambar 3.19. Terlihat bahwa konversi terjadi pada saat transisi turun dari sinyal clock.



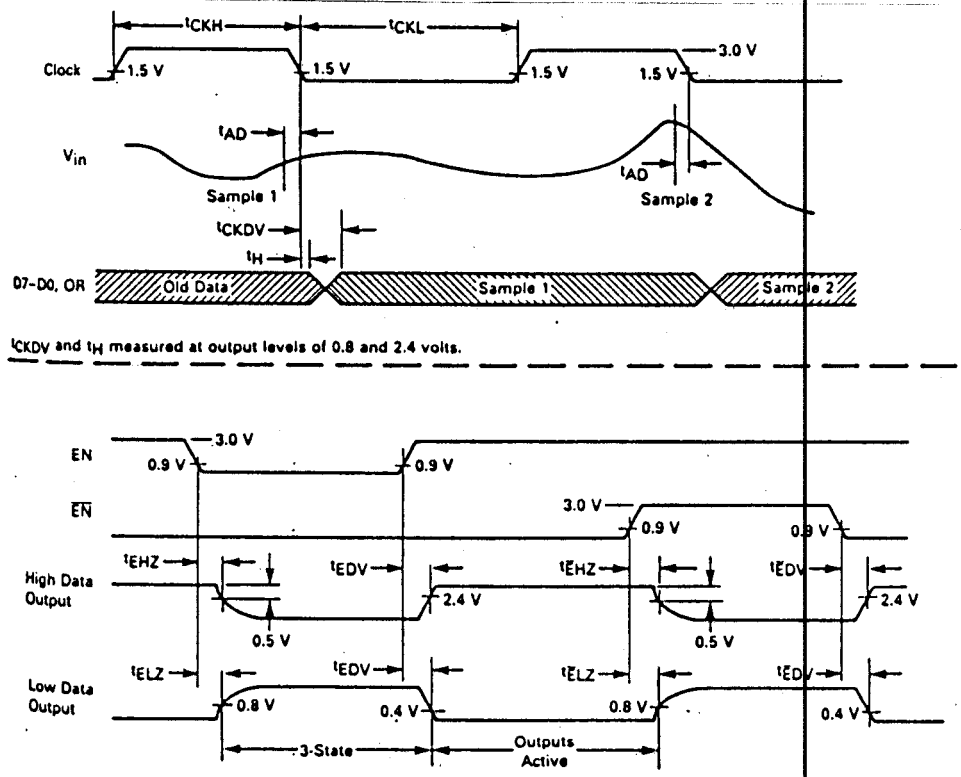
Gambar 3.19 Timing diagram MC10319

Rangkaian tegangan referensi

Tegangan referensi dari ADC menentukan level sinyal maksimum yang dapat dikonversi oleh ADC. ADC MC10319 mempunyai dua buah tegangan referensi, yaitu VRT untuk referensi positip dan VRB untuk referensi negatip.

Dari data ADC MC10319 diketahui kemampuan ADC untuk mengkonversi sinyal analog maksimum adalah 2 volt jika tegangan referensi positip VRT sebesar 2 volt dan tegangan VRB sebesar 0 volt.

Untuk ADC MC10319 direncanakan memakai tegangan referensi 2 volt. Rangkaian tegangan referensinya adalah seperti terlihat pada gambar 3.20.



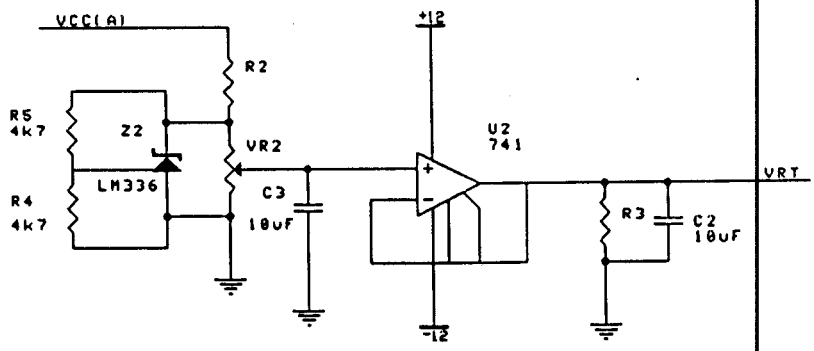
Gambar 3.19 Timing diagram MC10319

Rangkaian tegangan referensi

Tegangan referensi dari ADC menentukan level sinyal maksimum yang dapat dikonversi oleh ADC. ADC MC10319 mempunyai dua buah tegangan referensi, yaitu VRT untuk referensi positip dan VRB untuk referensi negatip.

Dari data ADC MC10319 diketahui kemampuan ADC untuk mengkonversi sinyal analog maksimum adalah 2 volt jika tegangan referensi positip VRT sebesar 2 volt dan tegangan VRB sebesar 0 volt.

Untuk ADC MC10319 direncanakan memakai tegangan referensi 2 volt. Rangkaian tegangan referensinya adalah seperti terlihat pada gambar 3.20.

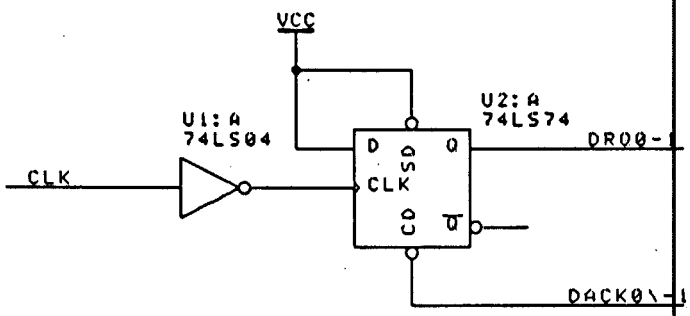


Gambar 3.20 Rangkaian tegangan referensi

ADC MC10319 adalah ADC flash yang hanya memerlukan satu sinyal clock untuk melakukan konversinya, yaitu pada transisi turun dari sinyal input clock. Untuk itu ADC ini sangat cocok jika diset pada mode *free running*, yaitu ADC diberi input sinyal clock dengan frekuensi tertentu (frekuensi sampling) sehingga secara terus menerus akan melakukan proses konversi tanpa mengindahkan siap atau tidak komputer menerima data. Karena ADC secara terus menerus mengirim data maka kontrol pengambilan data direncanakan melalui buffer yang menghubungkan modul ADC dengan komputer.

Perencanaan selanjutnya adalah hubungan dengan DMA Controller 8237-5. Karena DMA Controller menganggap tiap channel hanya berhubungan dengan satu peralatan I/O maka peralatan yang berhubungan dengan DMA tidak memerlukan dekoder alamat untuk akses peralatan I/O seperti pada umumnya. Satu-satunya penghubung antara peralatan I/O dengan DMA Controller adalah sinyal DREQ dan DACK\. Gambar 3.21 dibawah ini memberikan rangkaian pembangkit DMA request. Clock dari D flip-flop dihubungkan dengan clock frekuensi sampling yang telah diinverter, CLK\ sehingga DMA request akan

selalu hidup setiap transisi naik dari CLK\. Setelah komputer siap melayani permintaan DMA, maka jawaban berupa pulsa DACK3 diberikan. Pulsa ini dihubungkan dengan input clear D flip-flop sehingga jika aktif (low) akan mengakibatkan output D flip-flop kembali menjadi low. Akibatnya pulsa DREQ berakhir.

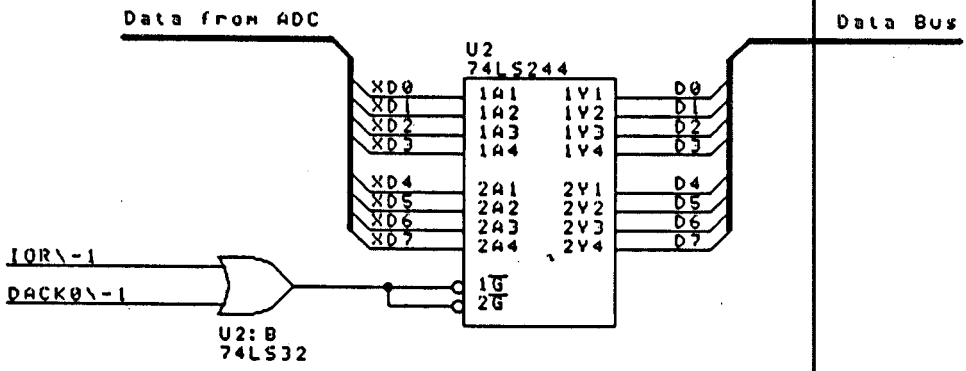


Gambar 3.21 Rangkaian pembangkit permintaan transfer DMA

Input D flip-flop adalah sinyal clock yang diinverter karena diinginkan permintaan transfer data dilakukan segera setelah konversi data terjadi. Proses konversi ADC MC10319 dilakukan pada transisi turun dari sinyal clock dan sedangkan input clock D flip-flop akan mengeluarkan sinyal DREQ pada transisi naik dari clock. Oleh karena itu sinyal clock perlu diinverter.

Selanjutnya setelah proses DMA telah dihidupkan dilakukan pembacaan data peralatan I/O dalam proses DMA. Untuk membedakan dengan sinyal baca yang dikeluarkan prosesor maka sinyal baca ini harus diikuti dengan aktifnya sinyal DACK\. Gambar 3.22 menunjukkan rangkaian pembacaan data peralatan I/O. Sinyal baca ini dihubungkan dengan buffer dari ADC MC10319 sehingga jika tidak ada proses DMA maka buffer akan berada dalam keadaan high impedance.

Dengan perencanaan seperti ini dipastikan ADC hanya mengirim data pada saat yang telah ditentukan saja sehingga tidak mengganggu proses dalam komputer.



Gambar 3.22 Rangkaian sinyal baca operasi DMA

BAB IV

PERENCANAAN PERANGKAT LUNAK

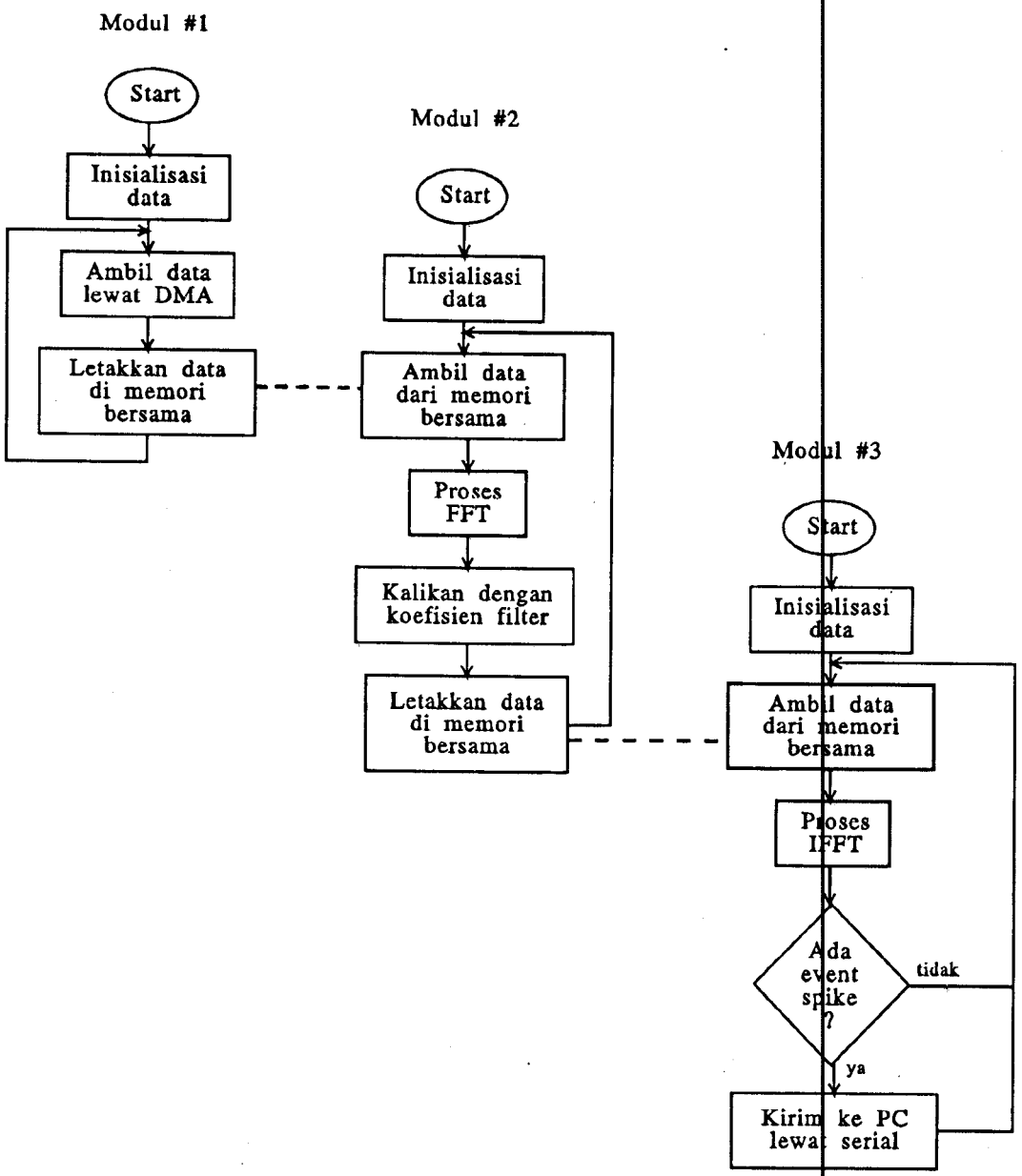
IV.1 Perencanaan perangkat lunak

Perangkat lunak yang direncanakan terdiri atas beberapa bagian, yaitu berupa pengambilan data melalui proses DMA, proses komunikasi antar modul, sistem filter digital FIR bandstop (*notch*) dan proses pengiriman data yang sudah difilter ke komputer melalui serial. Program ditulis dalam bahasa C dan assembly dan selanjutnya dikompile menggunakan Turbo C++ versi 1.0 dan Turbo Assembler versi 2.0.

Dalam pelaksanaannya program lebih banyak dikerjakan dalam bahasa C karena bahasa ini jauh lebih terstruktur dibanding bahasa Assembly dan lebih mudah untuk program berskala besar. Disamping itu hasil kompilasinya juga bisa digunakan untuk minimum system dengan mengadakan perubahan pada *start up code* compiler tersebut. Start up code ini berisi pengaturan segmentasi dan beberapa instruksi penting berhubungan dengan inisialisasi sistem perangkat keras.

Proses perangkat lunak ini disesuaikan dengan sistem perangkat keras yang telah direncanakan sebelumnya, yaitu dibagi menjadi tiga bagian dimana tiap bagian berada pada satu modul 8088. Bagian-bagian itu adalah proses pengambilan data lewat DMA pada modul pertama, proses FFT data dari modul pertama dan mengalikan data tadi dengan data filter telah di FFT dan mengirim hasilnya ke modul berikutnya pada modul kedua dan Modul terakhir berisi proses IFFT dan pengecekan event spike.

Apabila dideteksi ada event spike maka data dikirim lewat serial, jika tidak dilakukan proses modul ketiga dari awal. Diagram alur proses secara umum dapat digambarkan seperti pada gambar 4.1.



Gambar 4.1 Diagram alur proses akuisasi event spike

Garis putus-putus menggambarkan komunikasi antar modul yang dilakukan melalui memori bersama. Untuk mengetahui apakah data sudah dibaca atau data merupakan data baru maka digunakan suatu flag (*semaphore*) yang menunjukkan status tersebut. Flag besarnya satu byte dan letaknya di akhir blok data.

Direncanakan lokasi data transfer antara modul pertama dengan modul kedua berada pada alamat absolut 22500H sampai 24FFFH dan lokasi data antara modul kedua dengan modul ketiga berada pada alamat 20000H sampai 224FFH.

IV.2 Inisialisasi PIT

Proses akuisasi event spike membutuhkan suatu generator clock yang dapat diprogram agar mampu mengubah frekuensi sampling bila diperlukan. Frekuensi sampling ini harus akurat dan stabil terhadap noise yang ada karena kevalidan informasi yang diperoleh sangat bergantung pada frekuensi sampling tersebut.

Agar proses pengambilan data dapat berlangsung maka PIT harus terlebih dahulu diprogram pada suatu mode tertentu sehingga mengeluarkan gelombang kotak dengan frekuensi yang diinginkan.

PIT diprogram untuk bekerja pada mode *square wave*. Channel yang digunakan adalah counter 0. Direncanakan PIT mengeluarkan gelombang kotak dengan frekuensi 200 Hz. Dengan input clock ke counter 0 diambil sekitar 4.77Mhz / 128 = 37.23 KHz maka pembagi clock counter 0 adalah 186.15 dan dibulatkan menjadi 186. Keadaan lain yang diset adalah pembagi yang diambil besarnya 1 word dan paling tinggi adalah 65535 (FFFFH).

Dengan keadaan seperti diatas maka kontrol word yang dikirimkan adalah 36H.

Programnya adalah sebagai berikut :

```
asm   mov   dx,303h
asm   mov   al,036h
asm   out   dx,al
asm   mov   dx,300h
asm   mov   ax, 186
asm   out   dx,al
asm   xchg  al,ah
asm   out   dx,al
```

IV.3 Persiapan proses DMA untuk pengambilan data

Setelah PIT diset untuk mengeluarkan frekuensi sampling yang telah ditentukan maka selanjutnya proses DMA perlu diinisialisasi. Agar dapat mengambil data hanya pada saat yang telah diinginkan maka DMA Controller harus diset pada mode autointialize disable, sehingga setiap kali selesai melakukan transfer maka channel yang bersangkutan akan reset dan membutuhkan inisialisasi lagi jika dikehendaki proses transfer DMA kembali.

Berdasarkan pertimbangan tersebut maka DMA Controller diset pada channel 0, mode *non-autoinitialize*, single transfer dan blok data yang ditransfer adalah sebesar 1 kilo byte.

Dibawah ini adalah program untuk melakukan inisialisasi DMA Controller tersebut.

```
      asm   mov   al,04             // disable dma controller
      asm   out   COMMANDREG,al
/* ------initialize and start dma-------- */
      asm   mov   al,01000100b      // mode untuk channel 0
                  // single mode,address increment
                  // autoinitialization disable
                  // write transfer
      asm   out   MODEREGISTER,al
      asm   mov   al, byte ptr hi_addr
      asm   out   80H,al            // set page register
      asm   xchg  ax,ax
      asm   out   CLEARFF,al        // clear byte pointer F/F
      asm   mov   ax, word ptr lo_addr
      asm   out   0, al             // menentukan address DMA
                  // channel 0
```

```
asm    mov    al,ah
asm    out    0, al
asm    out    CLEARFF,al          // clear byte pointer F/F
asm    mov    ax,03FFH
asm    out    01,al               // menentukan counter DMA
                  // channel 0
asm    mov    al,ah
asm    out    01,al
```

Setelah menjalankan program diatas maka DMA Controller siap melakukan proses transfer DMA.

Modul ADC DMA yang direncanakan bekerja dalam mode *free running* seperti yang telah dijelaskan sebelumnya sehingga proses konversi data dan permintaan transfer DMA terus berlangsung. Untuk menghindari hal-hal yang tidak diinginkan maka proses DMA diprogram berlangsung pada saat-saat yang telah ditentukan.

Program dibawah ini bertugas mengaktifkan DMA Controller dan menunggu DMA sampai selesai melakukan transfer.

```
      asm     mov     al,00001110b     // mask all but chan. 0
      asm     out     0Fh,al
      asm     mov     al,00000000b     // enable controller
      asm     out     COMMANDREG, al
      asm     xchg    ax,ax

not_tc:
      asm     in      al,COMMANDREG
      asm     and     al,00000001b
      asm     cmp     al,1             // has dma at chan. 0 finished?
      asm     jnz     ok
      asm     jmp     not_tc

ok:;
```

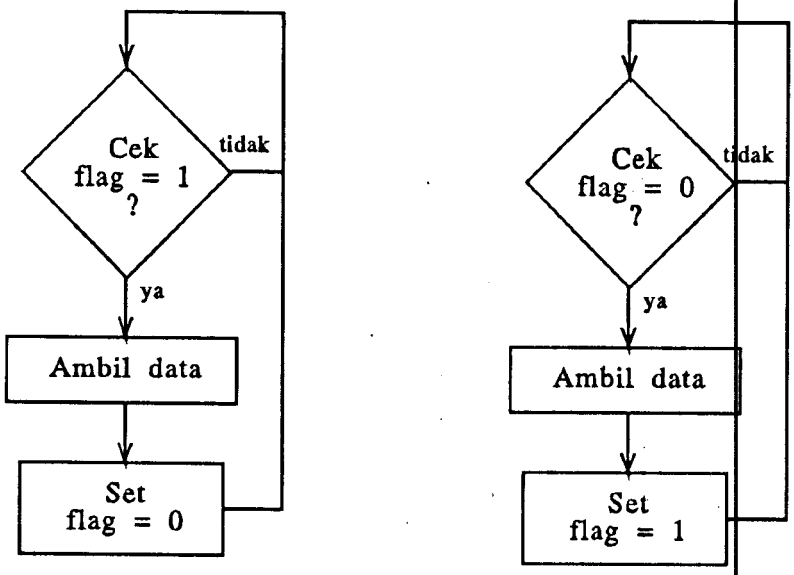
IV.4 Komunikasi antar modul melalui memori bersama

Komunikasi antar modul dilakukan jika diinginkan transfer data dari satu modul ke modul lainnya. Karena sistem yang dibuat tidak ada kemungkinan langsung transfer data antar modul maka langkah yang ditempuh adalah melalui

memori bersama.

Jika suatu blok data sudah dibaca dari memori bersama maka flag blok yang bersangkutan diset satu (01h) oleh modul yang membaca. Dan sebaliknya jika suatu modul menulis satu blok data baru di memori bersama maka isi flag blok data bersangkutan akan direset nol (00h).

Agar tidak terjadi penulisan data baru sebelum data lama dibaca atau sebaliknya, maka dilakukan pengecekan pada awal proses pembacaan atau penulisan dan pengubahan flag dari satu ke nol atau sebaliknya jika modul telah melakukan proses pembacaan atau penulisan. Diagram alurnya adalah seperti pada gambar 4.2.



Gambar 4.2 Diagram alur penulisan dan pembacaan memory bersama

Adapun programnya dalam bahasa assembly adalah sebagai berikut:

```
not_avail:;
      asm   mov   al, byte ptr [bx] // check flag = 1 ?
      asm   cmp   al, 1
      asm   jnz   not_avail
```

```
/* critical section goes here */
      asm   shr   cx,1          // change type from byte to word
      asm   cld
      asm   rep
      asm   segds
      asm   movsw         // baca blok data

    /* frees shared memory */
      asm   mov   byte ptr [si], 0  // set flag = 0
      asm   pop   ds
```

Dengan metode pembagian memori bersama menjadi beberapa bagian tersebut diatas menyebabkan satu lokasi memori bersama hanya bisa diakses oleh dua modul paralel. Ini mempermudah pengontrolan atas penulisan yang tidak dikehendaki saat satu modul memasuki daerah kritis (*critical section*).

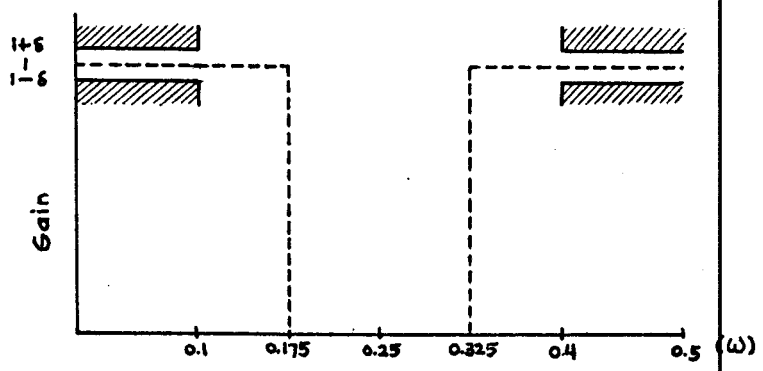
IV.5 Perencanaan filter digital FIR

Dalam perencanaan tugas akhir ini akan digunakan bandstop (*notch*) filter digital *Finite Impulse Response* (FIR) menggunakan window Kaiser. Spesifikasinya adalah sebagai berikut :

- Frekuensi sampling (f_s) : 200 Hz
- Ripple passband (A_p) : 10^{-4}
- Penguatan pada passband : 1
- Mempunyai pelemahan pada stopband : 80 dB (560 dB/dekade)
- Mempunyai respon frekuensi ideal sebagai berikut :

$$H(e^{j\omega}) = \begin{cases} 1 & \textit{untuk} \quad 0 \leq |\omega| \leq 0.175 \\ 0 & \textit{untuk} \quad 0.175 < |\omega| < 0.325 \\ 1 & \textit{untuk} \quad 0.325 \leq |\omega| \leq 0.5 \end{cases}$$

- Gambar respon frekuensi idealnya adalah seperti pada gambar 4.3.



Gambar 4.3 Respon frekuensi ideal filter FIR yang direncanakan

Setelah spesifikasi ditentukan maka selanjutnya dicari parameter α, D dan orde filter *(N)* menggunakan rumus dibawah ini :

$$D = \frac{A_a - 7.95}{14.36}$$

$$\alpha = 0.1102(A_a - 8.7)$$

$$N \geq \frac{\omega_s D}{B_t} + 1$$

Untuk orde filter diambil harga N ganjil terkecil dari persamaan diatas.

Setelah orde filter diperoleh maka langkah selanjutnya adalah mencari $\omega_k(n)$ dengan rumus window Kaiser sebagai berikut :

$$\omega_k(n) = \begin{cases} \dfrac{I_o(\beta)}{I_o(\alpha)} & |n| \leq \dfrac{N-1}{2} \\ 0 & \textit{lainnya} \end{cases}$$

$$\beta = \alpha \sqrt{1 - \left(\frac{2n}{N-1}\right)^2}$$

$$I_o(x) = 1 + \sum_{k=1}^{\infty} \left[\frac{1}{k!} \left(\frac{x}{2} \right)^k \right]^2$$

Untuk harga $I_o(x)$ yaitu fungsi Bessel orde ke nol tipe pertama dapat dicari dengan menggunakan prosedur dibawah ini :

```
float Io(float x)
{
    float Ds, D, Sum, compar;

/* Data initialization */
    Sum = 1;
    Ds  = 1;
    D   = 0;

    do {
        D  += 2;
        Ds  = Ds*x*x/(D*D);
        Sum += Ds;
        compar = Ds - 0.2e-8*Sum;
    } while(compar >= 0.0);

    return (Sum);
}
```

Selanjutnya setelah harga $\omega_k(n)$ didapat maka dicari persamaan $h(n)$ yang dikonversi dari persamaan respon frekuensi ideal filter bandstop $H(e^{j\omega})$. Persamaan $h(n)$ untuk bandstop filter diperoleh :

$$h_n = \frac{1}{\pi n} \sin \omega_{c1} n + \frac{1}{\pi n} \left(\sin \omega_{c1} n + \sin \frac{\omega_s}{2} n - \sin \omega_{c2} n \right)$$

Koefisien filter bandstop adalah perkalian dari $\omega_k(n)$ dengan $h(n)$ dengan n orde filter.

IV.6 Metoda filter data menggunakan FFT dan IFFT

Konvolusi merupakan bentuk langsung dari filter digital FIR. Namun pemrosesan filter FIR langsung dengan menggunakan konvolusi akan memakan banyak perkalian dan penjumlahan. Dengan konvolusi langsung maka besar perkalian yang dilakukan adalah sebesar $M*(N - M + 1)$[1] dimana N adalah jumlah data input dan M adalah panjang filter FIR.

Jika input data adalah sebesar 1024 data dan orde filter adalah 35 maka dengan konvolusi biasa besar perkalian adalah 1218560 kali, sedangkan jika menggunakan proses FFT IFFT besar perkalian adalah 45056 kali.

Berdasarkan pertimbangan diatas maka digunakan proses FFT untuk menghitung proses filter FIR.

Dasar utama penggunaan FFT dalam filter adalah teori konvolusi yang menyatakan bahwa konvolusi pada daerah waktu adalah sama dengan perkalian pada daerah frekuensi. Dengan demikian maka output filter dapat diperoleh dengan menggunakan urutan sebagai berikut :

1. Dengan menggunakan FFT diperoleh *H(k)* dalam frekuensi domain dari respon impuls h(n). Dalam hal ini *h(n)* adalah berupa koefisien filter FIR.
2. *X(k)* pada daerah frekuensi juga diperoleh dari input data *x(k)* dengan menggunakan FFT.
3. *X(k)* kemudian dikalikan dengan *H(k)* untuk memperoleh output *Y(k)*

---

[1]Embree, Paul M., *C Language Algorithms for Digital Signal Processing*, Prentice Hall International Inc., hal. 286

dalam daerah frekuensi.

4. Selanjutnya *Y(k)* ditransformasi balik ke daerah waktu menggunakan proses IFFT untuk memperoleh output *y(n)*.

IV.7 Pengiriman data melalui serial

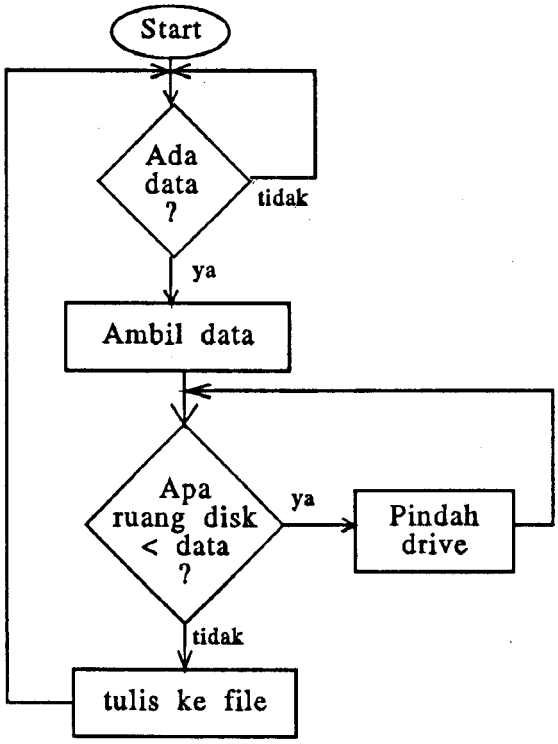
Pengiriman data lewat komunikasi serial dibutuhkan agar data yang telah difilter sebelumnya dapat diproses lebih lanjut di komputer yang mempunyai fasilitas lebih lengkap.

Agar data yang dikirim nilainya tertentu sehingga dapat dideteksi adanya kesalahan dalam transfer maka data sebelum dikirim dikonversikan lebih dulu menjadi kode ASCII-nya. Jadi misalkan data yang akan dikirim adalah 0FAH maka yang akan dikirim adalah 046H (kode ASCII untuk F) dan 040H (ASCII untuk huruf A). Dengan demikian data yang dikirim dapat dipastikan berkisar antara 30H sampai 46H, sehingga jika ada byte diluar harga diatas dapat dicek apakah merupakan *control byte* atau data salah (error). Control byte adalah byte dengan nilai khusus dan mempunyai tugas menginformasikan penerima data untuk melakukan suatu tugas.

Dalam perencanaan ini dipakai 2 byte sebagai control byte, yaitu satu untuk menginformasikan awal dari data dan satu lainnya berfungsi menutup blok data agar penerima data dapat menutup file segera setelah menerima control byte ini. Control byte awal data ditentukan 00H dan sebagai akhir data adalah 0FFH.

Selanjutnya penerima data dibuat selalu siap menerima data baru jika ada blok data baru datang dan terus demikian sampai ruang disk penuh. Jika isi disk

penuh maka program akan mengecek disk drive atau hard disk lain yang tersedia. Dengan cara demikian maka proses penerimaan data dapat berlangsung terus dan disket yang penuh selanjutnya dapat diganti dengan yang baru. Diagram alur dari proses penerima data adalah seperti pada gambar 4.4.



Gambar 4.4 Diagram alur proses penerima data

BAB V

PENGUJIAN

Pada bab ini akan dibahas pengujian terhadap implementasi dari perencanaan sistem yang telah dibuat sebelumnya.

V.1 Pengujian modul pembangkit frekuensi sampling

Dari perencanaan diinginkan frekuensi sampling adalah 200 Hz. Dengan menggunakan PIT dan input clock sebesar 37,286.92709 Hz maka diperoleh pembaginya adalah 186.4346355 dan dibulatkan menjadi 186. Akibat pembulatan itu frekuensi sampling yang dihasilkan menjadi 200.4673 Hz.

Dengan menggunakan penghitung frekuensi diperoleh input clock counter 0 adalah 37267 Hz dan frekuensi keluarannya adalah. Dari oscilloscope Hewlett Packard 2 channel 100 Mhz didapat interval 1 gelombang pulsa adalah $4.98.10^{-3}$; berarti frekuensinya adalah 200.8032 Hz. Ternyata hasilnya tidak begitu jauh dari harga yang direncanakan.

V.2 Pengujian modul paralel 8088

Setelah modul 8088 terakit, hal pertama yang dicek adalah keluarnya ALE (Address Latch Enable). Selanjutnya dicek apakah address decoder bekerja dengan baik, yaitu dengan berulang-ulang mengirim data tertentu ke suatu lokasi memori lokal. Hasilnya address decoder memori lokal bekerja dengan baik.

Setelah program dapat berjalan dengan baik pada suatu modul, maka selan-

jutnya dilakukan pengujian atas DMA Controller. Program yang digunakan untuk mengecek operasi DMA adalah program penulisan memori ke memori menggunakan DMA. Dari percobaan ternyata DMA Controller bekerja dengan baik.

Kerja dari arbiter 8289 dicek dengan penulisan secara berulang ke memori bersama. Program yang dipakai yaitu menulis suatu blok berisi data berurutan ke memori bersama. Selanjutnya data tersebut dibaca dari memori bersama. Jika data yang dibaca sama dengan data yang ditulis maka arbiter berjalan dengan normal. Pengujian berulang menunjukkan arbiter berjalan dengan normal sesuai perencanaan.

Hal terakhir yang diuji adalah operasi matematik dari *math co-processor* 8087. Dengan menggunakan suatu program sederhana menjumlahkan suatu bilangan float maka hasil yang diperoleh sama dengan yang didapat di komputer.

V.3 Pengujian unjuk kerja sistem modul paralel

Jika pada sub bab sebelum ini hanya mengecek operasi modul 8088 secara satu persatu maka dalam sub bab ini dilakukan operasi modul 8088 secara berbarengan (paralel). Pengujian dilakukan dengan menggunakan sebuah stopwatch digital dengan ketelitian sampai 1/100 detik. Dengan menggunakan metode penghitungan seperti ini hasil yang diperoleh memang tidak presisi, namun pengukurannya tidak sulit dan hasilnya masih dapat diandalkan.

Pengujian pertama adalah pengujian pengiriman data secara serial. Modul pertama berfungsi sebagai penyedia data yang akan dikirim dan modul kedua mempunyai tugas mengirim data tersebut secara serial. Ini bertujuan untuk

menghitung lama proses pengiriman data serial atau waktu yang dibutuhkan modul kedua untuk mengirim data dan tujuan kedua adalah mengecek apakah komunikasi antar kedua modul tersebut berjalan baik. Dari hasil file yang diterima komputer ternyata hasilnya sesuai yang diinginkan dan dari pengukuran yang berulang diperoleh lama transfer data adalah sekitar 19.5 detik. Selanjutnya pengujian yang sama diulangi terhadap modul ketiga.

Pengujian kedua adalah implementasi proses transformasi FFT dari data yang diperoleh. Modul pertama berisi program pengambilan data melalui DMA dan proses transformasi data ke daerah frekuensi menggunakan FFT. Modul kedua berisi proses pengiriman data ke komputer. Ternyata hasil yang diperoleh adalah seperti yang diharapkan, yaitu proses FFT bekerja dengan baik.

Pengujian selanjutnya adalah implementasi proses perkalian filter dengan data dalam daerah frekuensi dan proses IFFT. Jadi modul pertama berisi pengambilan data dan proses FFT; modul kedua berisi proses yang akan diuji tersebut diatas dan modul ketiga berisi pengiriman data serial tanpa ada pengecekan event. Hasil yang diperoleh adalah selang waktu dari pertama start sampai modul ketiga mengirim data adalah sekitar 12 detik (delay laten) dan selama waktu tersebut pengambilan data sempat dilakukan sebanyak 3 kali. Setelah modul ketiga mengirim data maka pengambilan data oleh DMA dilakukan setiap kali modul ketiga selesai mengirim data yaitu sekitar 19.5 detik. Disini terlihat interval antara pengambilan data menjadi lebih panjang karena adanya proses pengiriman data lewat serial.

Tabel 5.1 menunjukkan pengukuran waktu proses tiap modul dengan

pengiriman blok data lewat serial secara terus menerus tanpa pengecekan event spike.

Tabel 5.1 Pengukuran waktu proses tiap modul tanpa pengecekan event spike

Modul	Waktu proses	Waktu tunggu
I	8.5 detik	11 detik
II	3.8 detik	15.7 detik
III	19.5 detik	---

Terlihat proses terlalu banyak melakukan kegiatan menunggu proses lain selesai (proses mengirim data serial) sehingga dapat dipastikan proses ini mempunyai efisiensi tiap modul rendah seperti yang akan dihitung kemudian.

Selanjutnya pengujian dilakukan jika pengiriman data dilakukan hanya ada event spike saja. Hasilnya adalah seperti pada tabel 5.2.

Terlihat masih ada waktu tunggu yang lama saat tidak ada event, yaitu pada modul II dan III (lebih dari dua kali waktu proses modul yang bersangkutan). Ini karena pembagian task yang tidak seimbang; modul I lebih lama dari kedua modul lainnya.

Dengan mengadakan perubahan pada pembagian proses maka waktu tunggu modul berkurang cukup banyak seperti pada tabel 5.3. Hasilnya memang cukup baik, yaitu interval antara pengambilan data singkat yaitu sekitar 0.5 detik dan memungkinkan hampir semua event spike dapat diambil.

Tabel 5.2 Pengukuran waktu proses tiap modul dengan pengecekan event

Modul	Ada event		Tidak ada event	
	Waktu proses (detik)	Waktu tunggu (detik)	Waktu proses (detik)	Waktu tunggu (detik)
I	8.5	11.5	8.5	---
II	3.8	16.2	3.8	4.7
III	20	---	0.5	8

Tabel 5.3 Pengukuran waktu proses tiap modul dengan pengecekan event setelah perbaikan proses

Modul	Ada event		Tidak ada event	
	Waktu proses (detik)	Waktu tunggu (detik)	Waktu proses (detik)	Waktu tunggu (detik)
I	5.25	14.75	5.25	---
II	3.5	16.5	3.5	1.75
III	20	---	3.5	1.75

V.4 Penghitungan efisiensi

Pertama perlu dicari terlebih dulu lama proses akuisasi data event spike dengan menggunakan satu prosesor, yakni pengambilan data, proses dan output lewat serial. Dari pengukuran pada modul I dengan menggunakan oscilloscope diperoleh waktu pengambilan data DMA adalah sekitar 5 detik; waktu proses (FFT data, perkalian data dengan koefisien data pada daerah frekuensi dan IFFT)

pada satu prosesor 8088 4.77 Mhz sekitar 6.9 detik dan waktu pengiriman data lewat serial adalah kira-kira 19.5 detik sehingga total waktu yang diperlukan satu prosesor untuk menyelesaikan prosesnya adalah sekitar 31.4 detik. Lama waktu ini bisa menjadi basis waktu lama untuk tiap paket data, sehingga untuk paket data lebih dari satu maka waktu yang diperlukan cukup dengan mengalikan basis waktu tersebut dengan jumlah paket data.

Dengan proses pengiriman data serial terus menerus setiap ada pengambilan data (tanpa pengecekan event spike) maka bagian proses terlama adalah pengiriman data serial tersebut, yaitu 19.5 detik.

Untuk pengolahan 100 paket data, total waktu yang diperlukan proses pipeline diperoleh sebesar 1950 detik, yaitu jumlah paket data ditambah jumlah prosesor dikurang satu dan dikali dengan waktu dari proses modul terlama seperti yang telah dijelaskan pada bab II. Jadi perkiraan efisiensi untuk proses 100 paket data adalah:

$$\eta \approx \frac{100 \times 31.4}{3 \times 102 \times 19.5} \times 100\%$$

$$\approx 52.62\%$$

Terlihat efisiensi tiap modul jika proses pipeline mengirim data terus menerus dengan menggunakan tiga prosesor adalah rendah yaitu sekitar 52%, karena banyaknya waktu tunggu yang dibuang oleh tiap modul yang berarti prosesor tidak melakukan tugas apa-apa. Untuk pengiriman data secara terus menerus lebih efisien dengan menggunakan dua prosesor karena proses terlama

masih sama yaitu pengiriman data lewat serial sehingga waktu tunggu yang timbul bisa dikurangi.

Bila pengiriman data dibatasi hanya jika ada event spike maka efisiensinya dihitung sebagai berikut :

1. Dihitung lama proses jika dikerjakan oleh satu prosesor dan diperoleh sekitar 12.4 detik.
2. Dicari proses terlama pada proses pipeline dan didapat pada modul I yaitu sekitar 5.25 detik.
3. Efisiensi untuk proses 100 paket data diperoleh sekitar :

$$\eta \approx \frac{100 \times 12.4}{3 \times 102 \times 5.25} \times 100\%$$

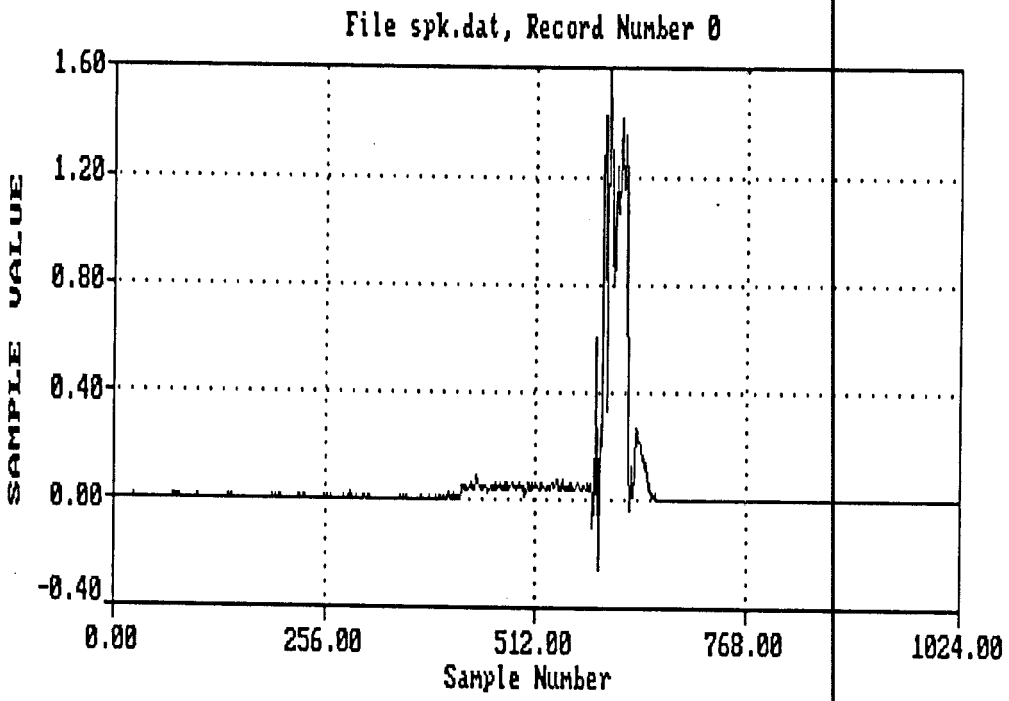
$$\approx 77.18\%$$

Selanjutnya tabel 5.3 menunjukkan hasil perhitungan efisiensi tiap modul dari proses pipeline terhadap jumlah paket data menggunakan 3 prosesor.

Tabel 1. Efisiensi tiap modul dari proses pipeline terhadap jumlah paket data tanpa event

Paket data	Efisiensi (%)
1	26.24
2	39.36
4	52.48
8	62.98
16	69.98
32	74.09
64	76.34
128	77.51
256	78.11
512	78.42

Dibawah ini adalah contoh hasil pengambilan output tegangan DC saat terjadi perubahan dari on ke off atau sebaliknya dari switch power supply DC Hewlett Packard.



Gambar 5.1 Contoh data event spike yang diperoleh

BAB VI

KESIMPULAN DAN SARAN

VI.1. KESIMPULAN

Kesimpulan yang dapat ditarik dari pembuatan tugas akhir ini adalah sebagai berikut :

1. Proses paralel secara pipeline sangat membantu dalam proses akuisasi event secara real time.
2. Perencanaan perangkat lunak dan pembagian tugas diantara modul prosesor sangat menentukan kecepatan dan efisiensi dari proses paralel pipeline.
2. Pemakaian Direct Memory Access (DMA) sangat cocok untuk aplikasi yang membutuhkan kecepatan tinggi dalam pengirimannya.
3. Co-processor 8087 sangat cocok dalam aplikasi yang banyak memakai perhitungan bilangan real. Penggunaan Co-processor ini memungkinkan program kalkulasi bilangan real dalam format .COM sehingga dapat dimasukkan dalam EPROM.
4. Dalam proses pipeline efisiensi tiap modul sangat bergantung pada jumlah modul yang bekerja paralel dan jumlah paket data yang diproses.
5. Proses FFT dan IFFT sangat membantu dalam mempercepat proses filter digital FIR dan juga pemakaian proses tersebut memudahkan pembagian tugas (task) diantara modul.

6. Pemakaian memori lokal mengurangi kesibukan pemakaian multibus yang akhirnya menaikkan kecepatan proses pembacaan ke memori bersama.
7. Frekuensi sampling yang stabil dan akurat sangat berpengaruh dalam pengambilan data yang benar.

VI.2 SARAN-SARAN

Berikut ini diberikan saran-saran yang diharapkan dapat berguna bagi penelitian ataupun pengembangan lebih lanjut dari tugas akhir ini.

1. Jika mungkin digunakan memori yang mempunyai kapasitas yang besar sehingga mampu mengolah data real yang besar. Dalam hal ini bisa digunakan memori dinamik.
2. Pengiriman data lewat serial merupakan penghalang terhadap kecepatan proses secara keseluruhan sehingga untuk pengembangan lebih lanjut perlu digunakan cara lain yang lebih cepat seperti penyimpanan secara langsung ke disket.

DAFTAR PUSTAKA

1. Antoniou, Andreas, ***Digital Filters: Analysis and Design***, Tata McGraw-Hill Publishing Company Ltd., New Delhi, 1980.

2. Axford, Tom, ***Concurrent Programming : Fundamental Techniques for Real-Time and Parallel Software Design***, John Wiley & Sons Ltd. England, 1989.

3. Cok, Ronald S., ***Parallel Programs for the Transputer***, Prentice Hall Englewood Cliffs, New Jersey, 1991.

4. Embree, Paul M., ***C Language Algorithms for Digital Signal Processing***, Prentice Hall International Inc.

5. Hall, Douglas V., ***Microprocessors and Interfacing : Programming and Hardware***, McGraw-Hill Book Co., 1986

6. Liu, Yu-Cheng, ***Microcomputer systems : The 8088/8086 Family : Architecture, Programming, and Design, 2nd Edition***, Prentice Hall of India, 1986.

7. Mano, M. Morris, ***Digital Logic and Computer Design***, California State University, Los Angeles.

8. Oppenheim, Alan V., ***Discrete-Time Signal Processing***, Prentice Hall International Inc.

9. _____, ***Microsoft Macro Assembler 5.1 : Programmer's Guide***, Microsoft Corp., 1987.

10. _____, ***Microsystem Components Handbook : Microprocessors Volume I***, Intel Corporation, 1985.

11. _____, ***Technical Reference for PC/XT System***, IBM Corp.

12. _____, ***Turbo C Version 2.0 : Reference Guide***, Borland International.

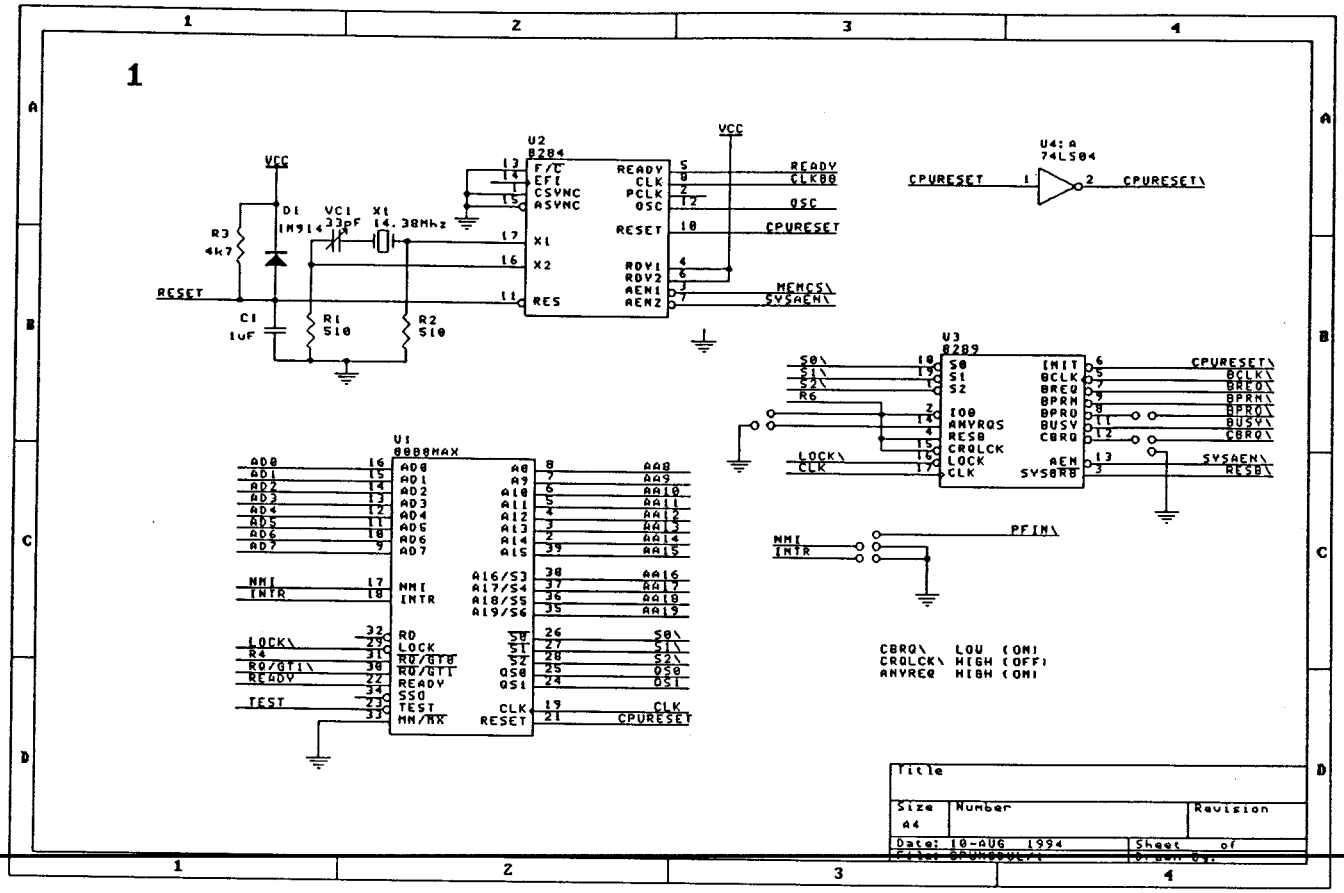


Diagram Skema Modul 8088 Paralel (Gambar 1 dari 9)

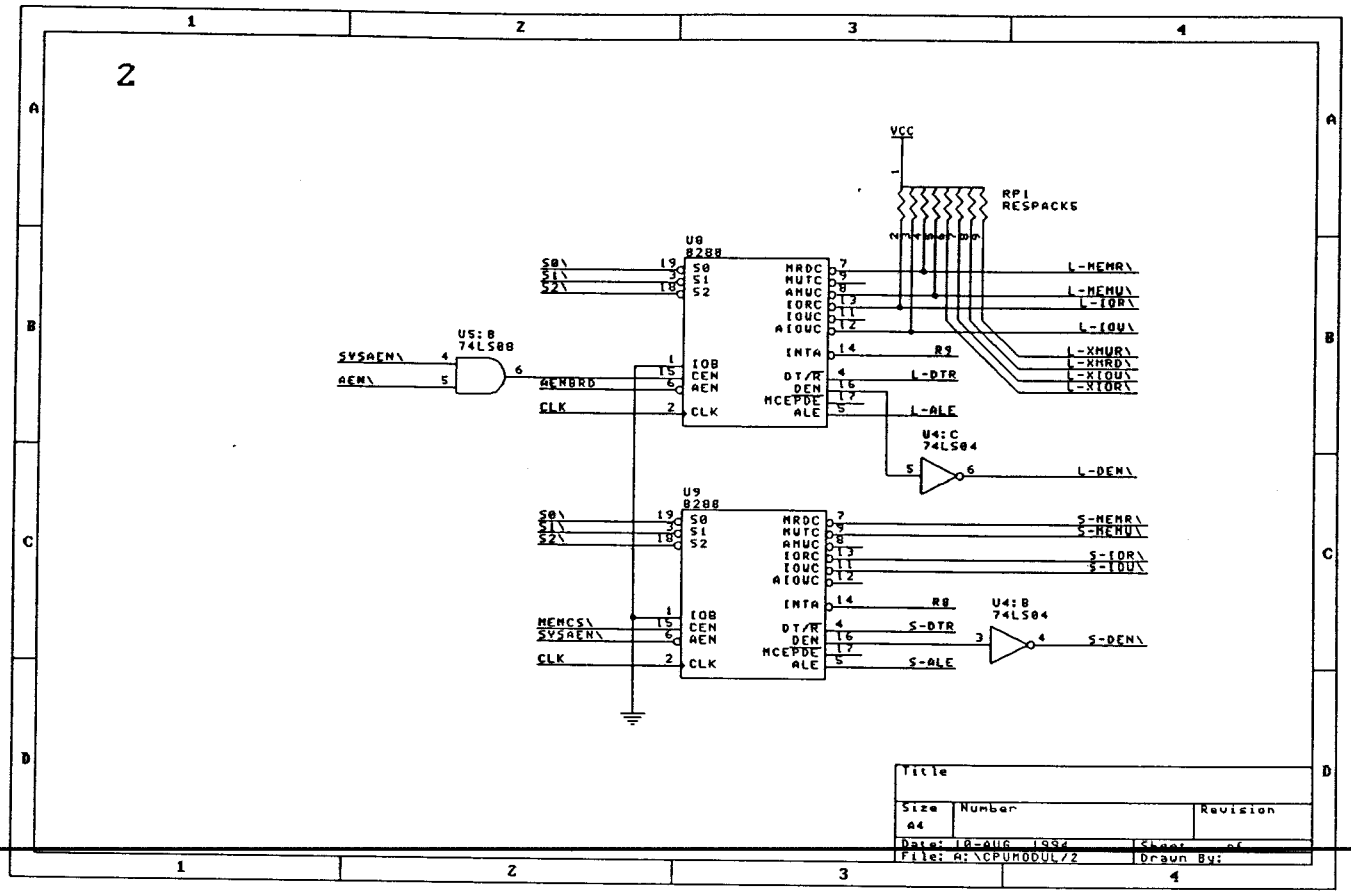


Diagram Skema Modul 8088 Paralel (Gambar 2 dari 9)

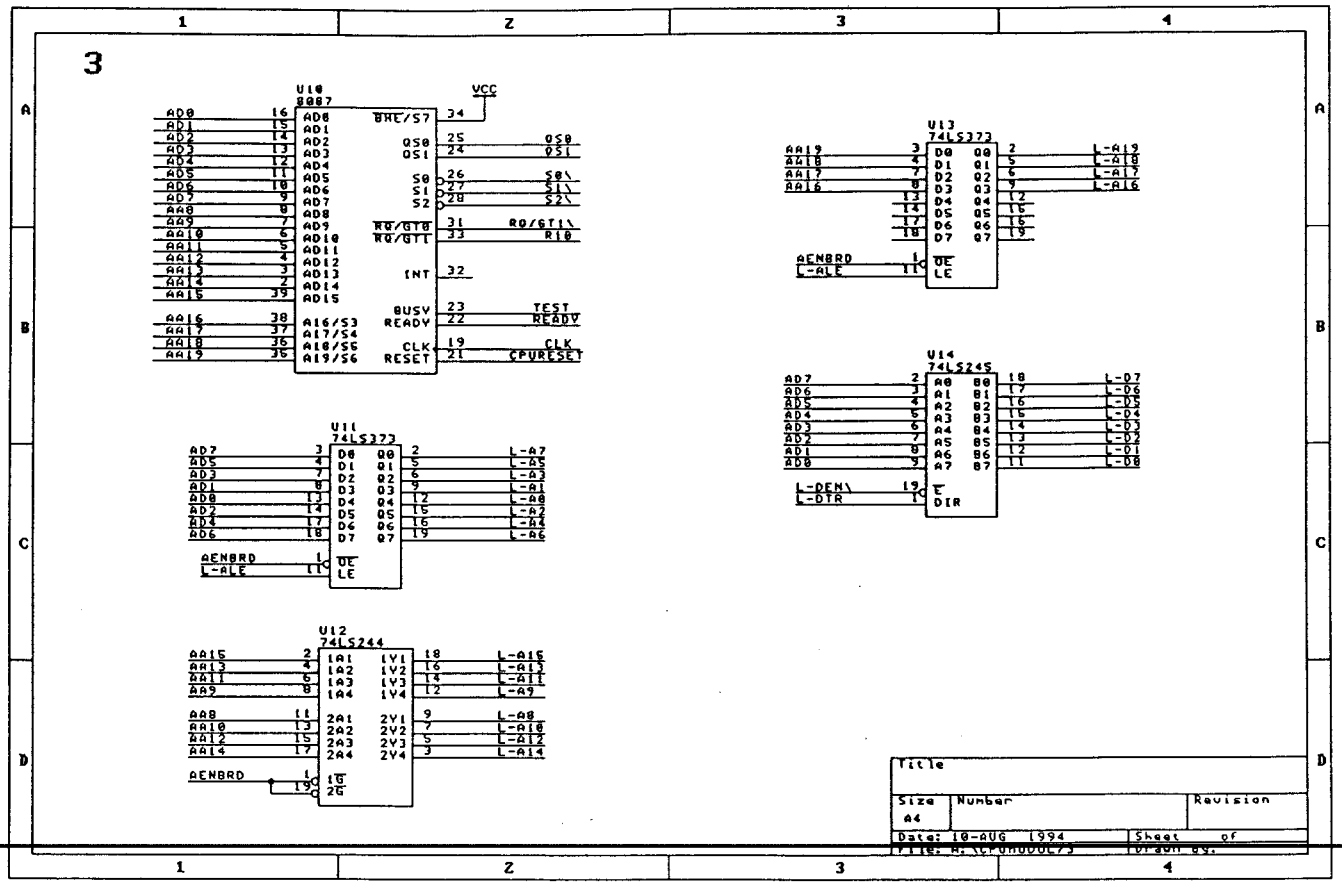


Diagram Skema Modul 8088 Paralel (Gambar 3 dari 9)

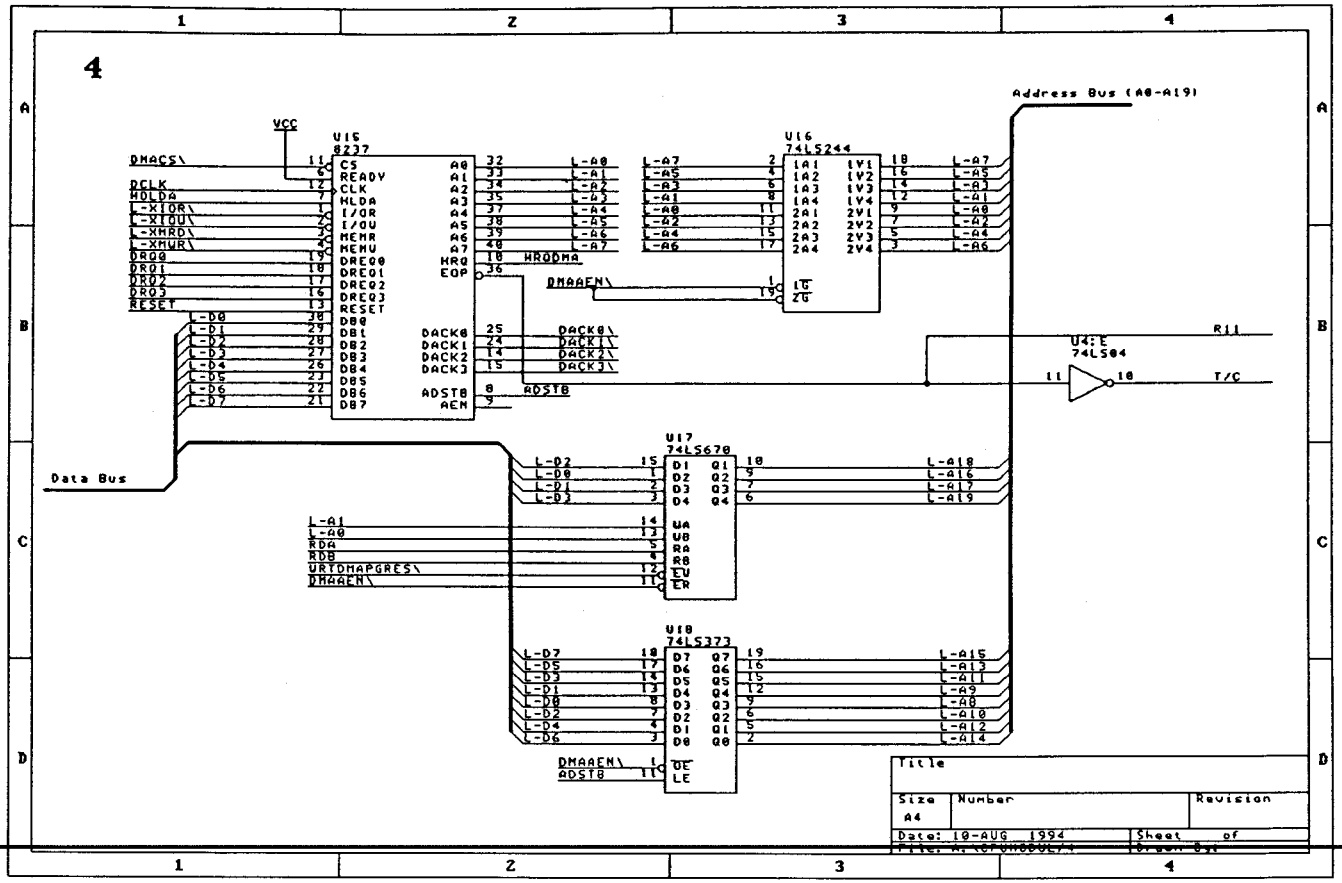


Diagram Skema Modul 8088 Paralel (Gambar 4 dari 9)

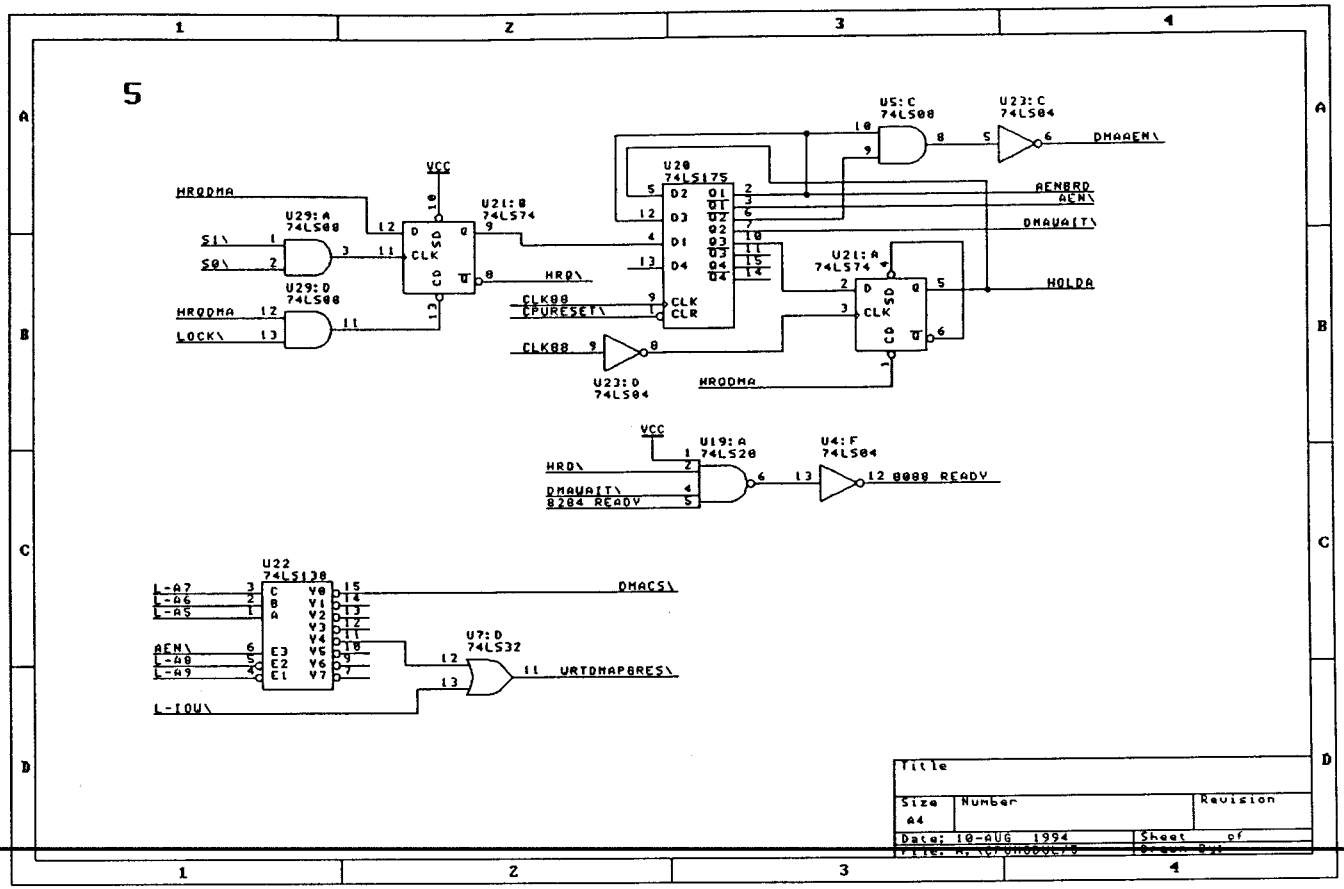


Diagram Skema Modul 8088 Paralel (Gambar 5 dari 9)

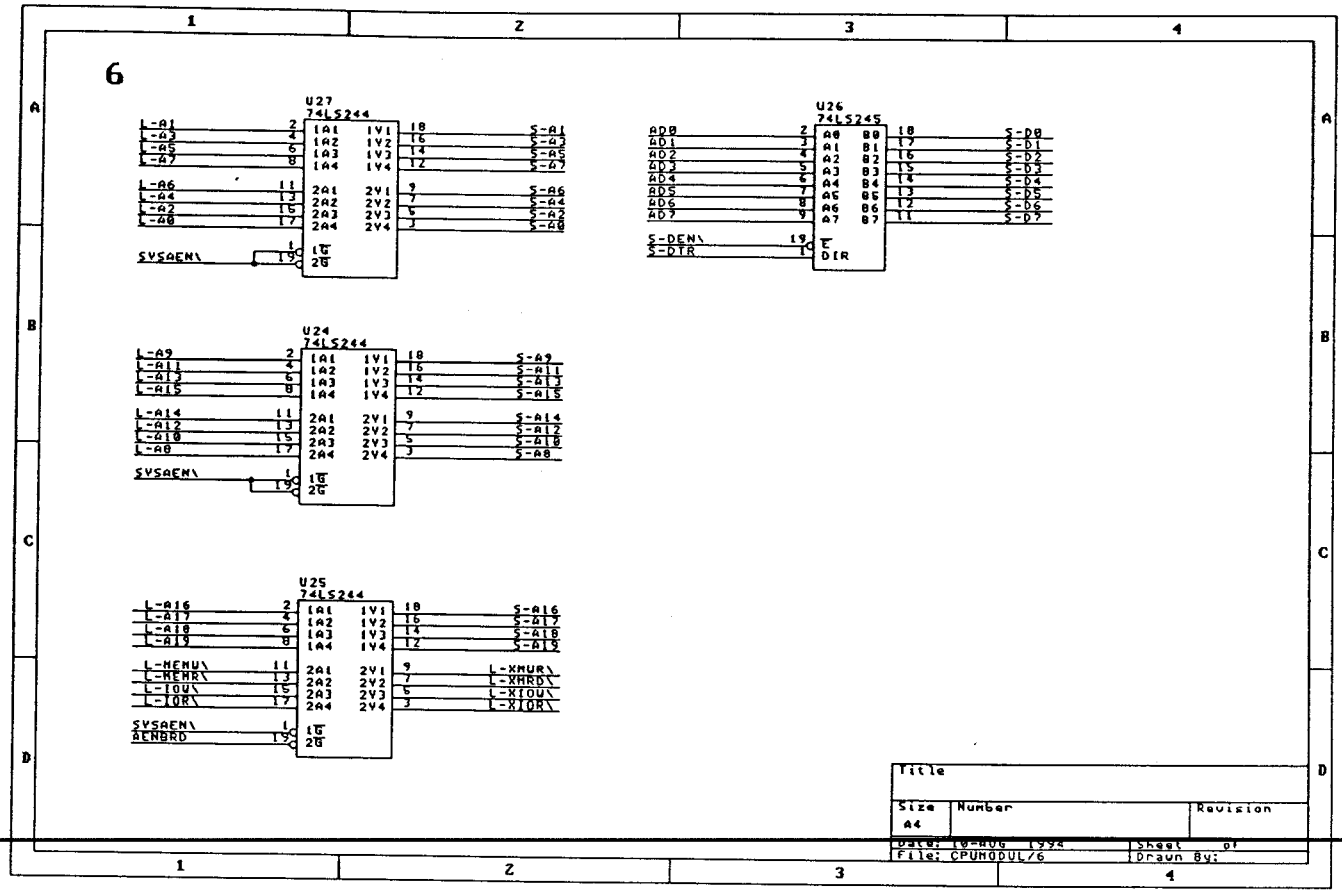


Diagram Skema Modul 8088 Paralel (Gambar 6 dari 9)

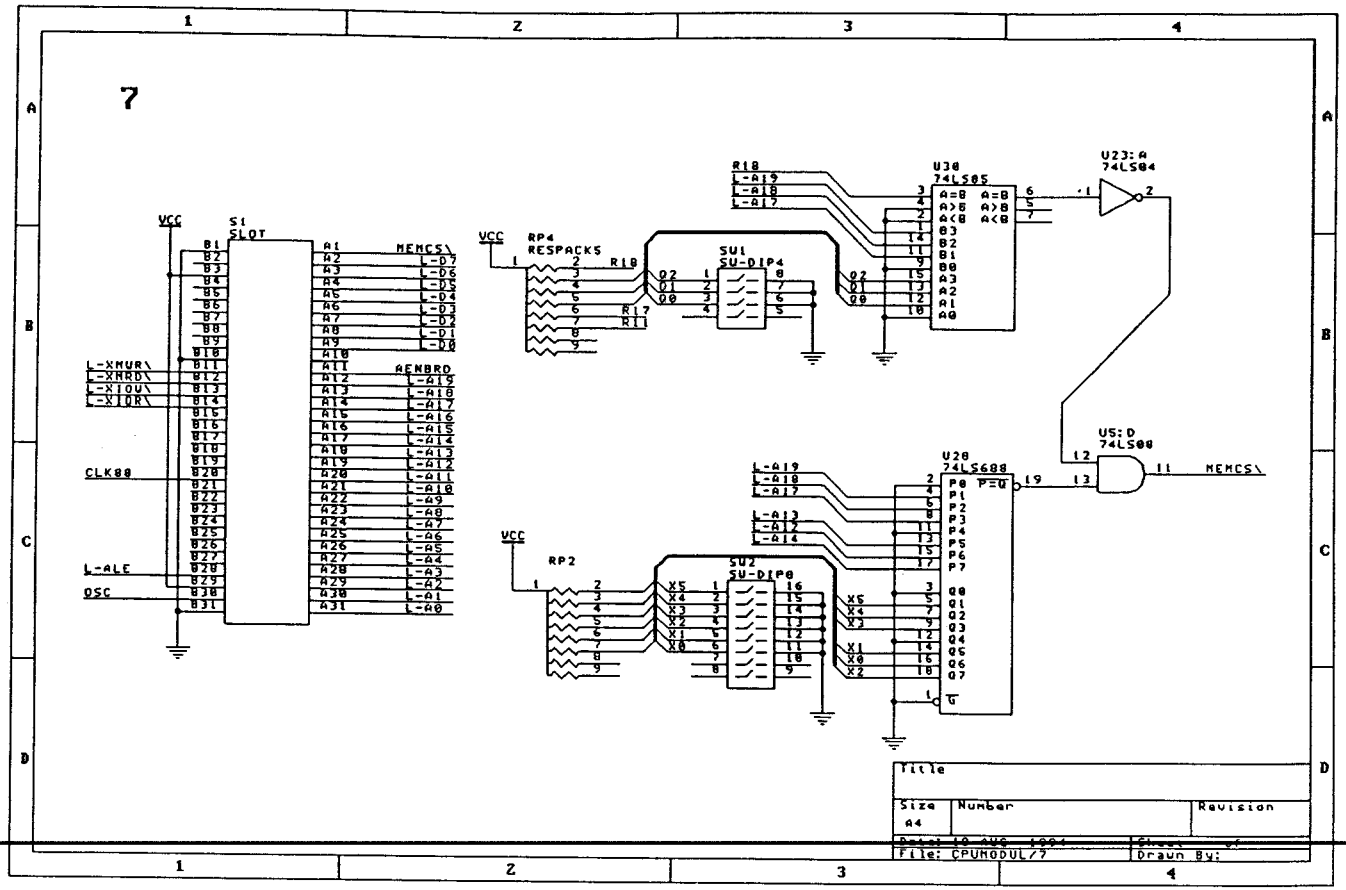


Diagram Skema Modul 8088 Paralel (Gambar 7 dari 9)

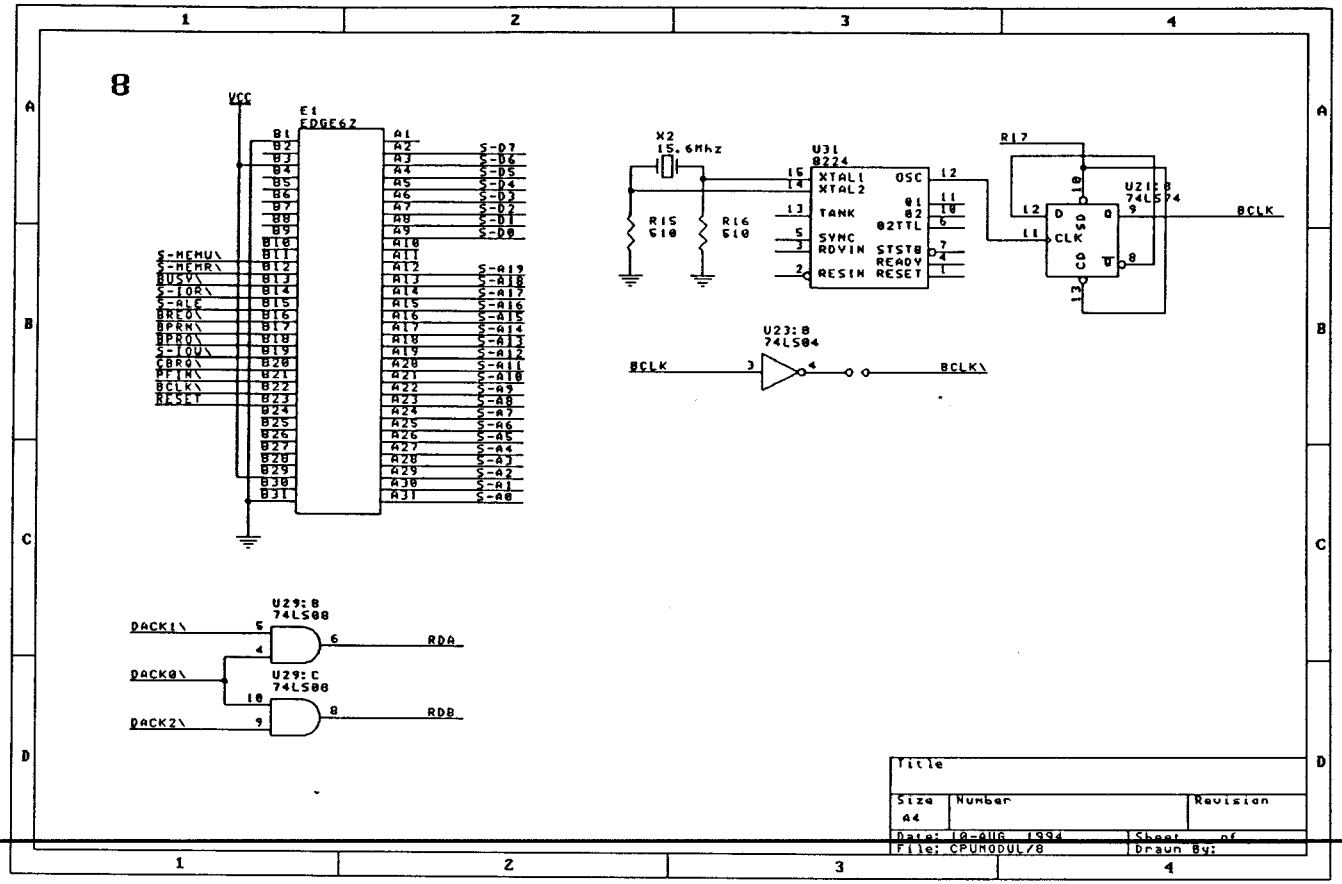


Diagram Skema Modul 8088 Paralel (Gambar 8 dari 9)

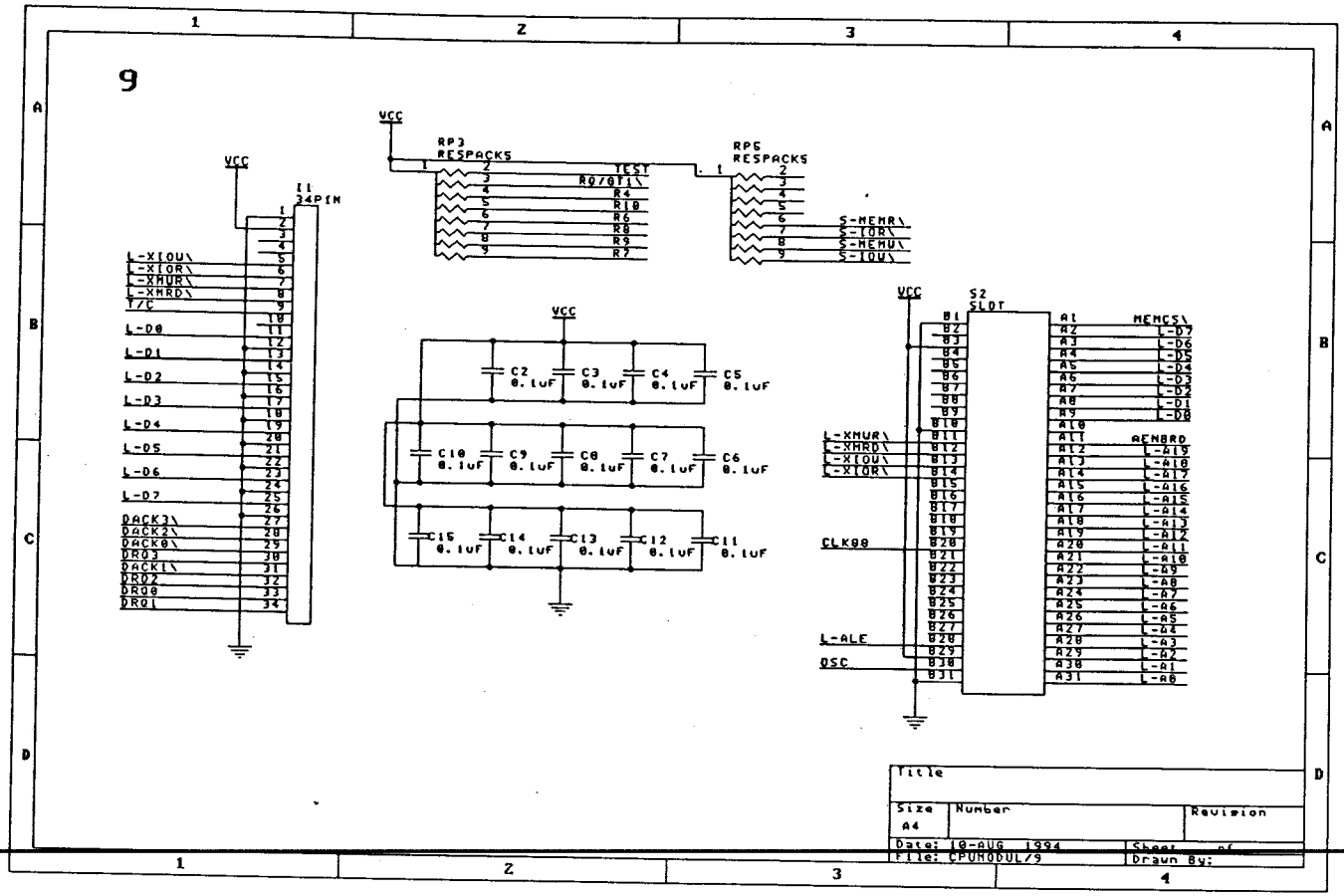


Diagram Skema Modul 8088 Paralel (Gambar 9 dari 9)

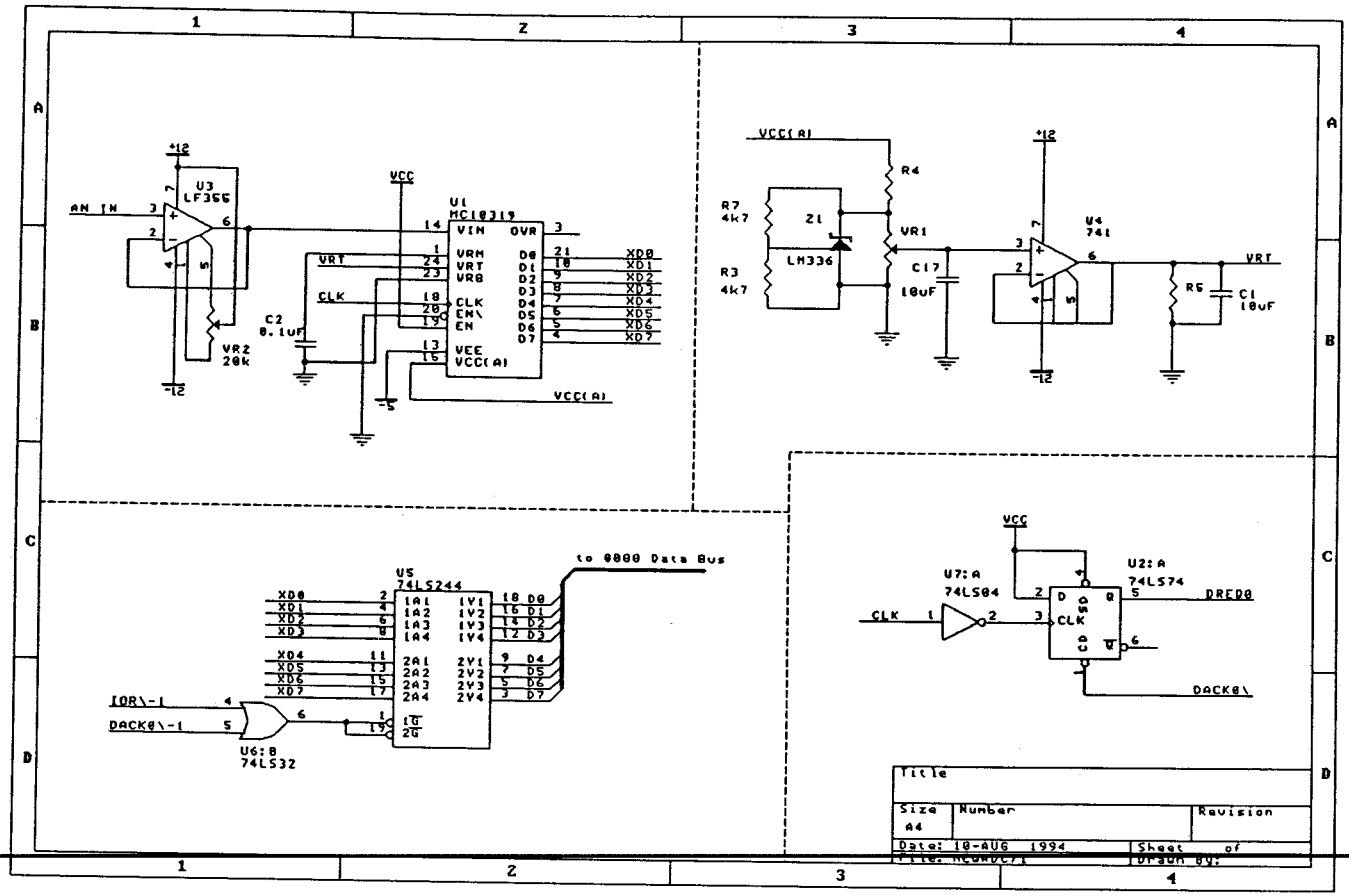


Diagram Skema Modul ADC lewat operasi DMA (Gambar 1 dari 1)